# IPSO FACTO

ISSUE #5
MARCH, 1978

(A publication of the Association of Computer Experimenters)

TABLE OF CONTENTS



Editor : Tom Crawford
Invaluable Assistants: Wayne Bowdish, Diane York, and all contributors to this issue.



All Newsletter correspondence should be sent to:
Tom Crawford,
50 Brentwood Drive,
Stoney Creek, Ontario,
CANADA L8G 2W8

## EDITOR'S REMARKS MAR. 26, 1978

I am pleased to report that, as of this date, our membership stands at 351 paid members. This includes 242 Canadians, 105 Americans, and 4 International members. I would like also to report that, out of this total membership, less than 35 people have contributed to the Newsletter (not counting letters to the Editor). I trust the remaining 90% are seriously considering a submission in the near future. Remember: you don't have to be an expert in your field to write for this Newsletter. All you need is an idea, a pencil and a piece of paper! You don't even have to have a working application; if you've been mulling over some idea, put it down on paper in the form of a proposal, and send it in. Perhaps someone can help you out with some practical suggestions; or perhaps they have already done it, and can tell you how to get yours working. Surely you haven't all packed your 1802s into a box on the shelf and forgotten about it! The prime purpose of this Newsletter is to communicate ideas between members, but you must put your ideas down on paper first, so get to it!

### MACHINE OR ASSEMBLY LANGUAGE?

The question of whether to document a program in machine language is a many-faceted one. On the one hand, a hex dump in machine language is a simple procedure, taking little time, and occupying only a small space. On the other hand, an assembly language listing, with comments and assembled machine code, is a lengthy procedure, requiring a large expenditure of time, and generally occupying a substantial amount of space. But consider the real purpose of writing a program down on paper. Generally, you wish to make it available to someone else, or even to yourself, at a later date. Unless your program is a trivial one (say 10 bytes or less), will you or anyone else be able to understand what it does, or how? Will your friends (or fellow ACE members) be able to modify your program to run with a different clock frequency, or different I/O devices? What if they load your program, and it doesn't work. Will they ever be able to figure out for themselves what's wrong? All of the above questions point to the need for comments on a program listing, to explain the details of a programs operation. Comments aren't difficult to produce; they usually reflect the thinking of a programmer as he designs a program, and gets it to work. All the programmer needs to do is write down a comment, each time he writes a machine instruction. Easy, right?

The use of assembly language allows the programmer to easily remember an instruction by its mnemonic rather than by its hex code. It also allows the use of labels for branch points and variable or fixed data locations. These labels can be easily found later, when you must re-assemble your program because you needed an extra instruction in the middle.

The biggest argument against the use of assembly language is said to be the requirement for an assembler program. This simply isn't true. Admittedly, an assembler is nice to have, but you can gain many of the benefits of assembly language when hand-assembling the

MACHINE OR ASSEMBLY LANGUAGE CONT'D

machine code, especially as your programs exceed the 10-20 byte size. Anyway, how many of you can really think in machine code when reading, say, a 256 byte program written by someone else? Think about it.

I think the best way to illustrate my point is with an example, and an excellent one has come to hand. Recently, two large circulation magazines, POPULAR ELECTRONICS and DR. DOBB'S JOURNAL, published an interesting monitor program for the 1802 (Ed, you were going to tell me why they both printed it?). Unfortunately (in my opinion), both of these magazines published only an uncommented hex dump listing of this program. How many of you, when faced with this type of program documentation, decided that it wouldn't be worth the effort? How many others tried it, but couldn't get it to work right? And finally, how many of you began to dis-assemble, and comment the program, in order to figure out how to get it working on your particular 1802? Wouldn't it have been nice to have all that information in the original listing? In this issue of IPSO FACTO, you will find a dis-assembly of Ed McCormick's monitor program. I want you to compare this form of documentation with the hex dump published by PE and DDJ. I hope this example makes my point.

SIMPLE APPLICATIONS

In order to maintain the involvement of those not (yet) interested in a general purpose computer system, we intend to put more emphasis on simple, fun-type applications, requiring only an 1802, 256 bytes of memory maximum, and perhaps a few bits and pieces of interface hardware. The first one is in this Issue: The 8 digit hex display for under $10. This simple application will be extended in the next issue, to become the display for a Real Time Clock. You will also find a speed control for those cheap 6 Volt DC motors, which requires only a power transistor, a door bell control based on your 1802, and a sump pump timer. How about you? Let's hear about your application: the simpler the better. Remember: the best way to learn is by doing.

I will apologize at this point for the delay in getting out Issue #4; we had a few printing problems. I expect we will soon be back on schedule.

NETRONICS, ANYONE?

Speaking of schedules, has anyone received a news bulletin from Netronic's Elf User's Club? I have had several people ask me this question. Their memberships, at $3 each, go back about 8 months. Perhaps someone from Netronics R&D, Ltd. would care to reply c/o this Editor?

## INTERNATIONAL POSTAL COUPONS

I have had a number of people send me International postal coupons for postage. These coupons cost the buyer in the U.S. 42¢, yet can only be redeemed for Canadian postage in the amount of 12¢ or 25¢ (depending which postal clerk I talked to). I suggest that a 190% to 400% mark-up is a little much. Cash or money order would be a better deal, I think. Please remember, though, that only Canadian postage can be used on letters mailed in Canada.

## AN 1802 HAM NET

A number of ACE members are also hams, so some members of our local amateur radio club will be attempting to set up a "net" to discuss common 1802-type problems in the near future. They will try to extend coverage as far as possible (interest already exists in British Columbia, Chicago, and Washington, D.C.!). Contact the following person for more information:

Brian Fox (VE3 EBF)
Fox Communications Ltd.
124-3 King St. W.
Stoney Creek, Ont.
Canada L8G 1J2
Phone (416)664-5433

## INTERESTING ARTICLES

Two articles in particular have been drawn to my attention. The first is a 3-part Video Display Unit construction article, published in Electronics Today International magazine, August-October 1976. This unit displays 8 rows of 32 characters each on a TV set or video monitor. Most of the circuitry is TTL, except for the character generator (a 2513) and the memory (2X2112's). PC Board layouts (single-sided) are included in the article. I would estimate the parts cost to be about $50. Several people are currently involved in building one of these units. I shall be reporting on their progress.

The second article can be found on pages 19 to 21 of the December 1977 Circuits and Systems Journal of the IEEE. Written by John Doyle of the University of Toronto, the article is entitled Microprocessor Interfacing. It shows several very simple interfacing techniques to allow connection of LED's, speakers, meters, potentiometer, 8 bit D-A's and op amps to your microprocessor. If you are looking for a copy of this Journal, I suggest you try the Science and Engineering Library at your local college or university.

## HARDWARE PAPER TAPE LOADER

Harley Shanko and Jorgen Munck

This design resulted from the authors' need for a simple means of generating a media for program storage and for entering these programs into their uC system. Each had a paper tape reader that was not yet in use, and a RCA 1802 uP based system. The dilemna was this: Isn't there some simple way to punch tape and be able

to load it without requiring software to do the ASCII-to-HEX conversion and HEX-pair packing into bytes?

Yes! Here is a relatively simple, low cost solution. It requires only 4½ CMOS IC's, exclusive of the input port. Tape preparation is very simple; in the extreme, only the object code need be punched, with some leader/trailer at each end. This design assumes a compatible interface exists at the paper tape reader, with +V = Logic 1, OV = Logic 0, for both sprocket and data signals.

## MEDIA FORMAT AND PREPARATION

It is assumed that a Teletype (TTY) or similar device is available to punch tape and to print hardcopy for a record of the tape contents. This hardcopy is helpful for verifying accuracy of the data on the tape, but is not an absolute necessity.

The data to be punched onto the tape is the object code (machine code) of your program(s). This article deals with hexidecimal (hex) coding only. Thus each byte will be represented on tape by two hex characters in ASCII coding. See Table I for a step-by-step tape generation procedure. This format is useable for both motor driven and manually-pulled tape readers; the authors have interfaced both types. Since the manual type requires a little additional 'formatting' for convenience, that format will be described. By experience it was found to be difficult to 'pull in' more than 256 bytes per 'pull'. Note that, at 10 characters/inch, that is about 4 1/3 foot 'stretch'. About one foot of trailer is used to follow each 256 bytes as a zone to 'stop on'. Otherwise, tape movement between successive pulls could cause erroneous data entry, with the o.1 inch tape hole spacing. All ASCII control characters (non-printable codes, column 0 and 1) are transparent to the loader. Several of these control characters are used advantageously; NUL's are used for the leader, trailer and between the 256 byte blocks; CR and LF are used to format the hardcopy. Figure 1 illustrates a representative listing of a formatted object code tape hardcopy.

## THEORY OF OPERATION

The loader logic consists of three sections: control, conversion, and input port.

The control section consists of two flip-flops and a two-input OR-gate. Figure 2 illustrates the loader timing and Figure 3 the logic. The OR-gate is used to 'detect' the column 3 and 4 ASCII-HEX codes. Full decoding for ASCII-HEX 0 - 9, and A - F codes is not implemented: and printable ASCII, columns 2 - 7, could be loaded, but since the object code will only be hex, this fact permits the simpler decoding. The presence of a valid code and the leading edge of the sprocket pulse sets F/F 1, removing the 'set' signal from F/F 2. This signal provides initialization to F/F 2. F/F 2 divides the sprocket timing by two: the positive transistions of F/F 2 $\overline{Q}$ line sets the DMA flip-flop, F/F 3, TRUE. F/F 3 Q output then pulls the DMA-IN line. When sensed by the

uP, an input port Enable is generated (S2 or SC1), resetting the DMA flip-flop and enabling the port 3-state gates. The register contents are then present on the data bus, and the uP writes the data into memory.

The conversion section consists of a 4-bit full adder and two 4-bit registers. Table II and its note describes the ASCII-to-HEX conversion. Simply stated, the 4 LSB ASCII code bits are ADDED to a value of zero or nine, depending on the state of input bit 7. When hex A - F (ASCII codes 31 - 46) are present will bit 7 be TRUE; then a value of 9 (binary 1001) is presented to the B-input of the adder. The resultant sum is the desired hex code. The adder outputs are clocked into the first register, and, simultaneously, that register outputs, containing the previous value, is clocked into the second register. This presents an 8-bit code to the input port.

The input port consists of eight 3-state gates (2/3 of the 80C97 and the output section of the 4076) and F/F 3, previously described. In an 1802 system, F/F 3 $\overline{Q}$ line connects to the DMA-IN line, with S2 or SC1 used for the input port enable signal. If the 1802 Interrupt line is not used, SC1 can be used; otherwise S2 is required; $\frac{1}{2}$ of a 4555 dual-decoder can be used to generate all four State decodes.

## MULTIPLE DMA-IN PORTS

A contention problem will exist if the tape loader is logic-ORed with a DMA-IN signal from another input device. The simplest solution is to add a switch to select which port shall receive the SC1 signal, in order to enable the output of that port to the data bus. Also, wire all ports' DMA flip-flop RESETs directly to the SC1 signal. What this provides is a reset to all DMA flip-flops, whether that port is selected or not. The reason for this is because on power up one or more DMA flip-flops will come up set, and will produce a continuous DMA condition on the ORed DMA-IN line, until it gets reset. Figure 4 shows a typical arrangement.

## TABLE I

### TAPE PREPARATION

1. Turn on TTY and set to LOCAL mode.
2. Punch about one foot of leader (press HERE IS key several times).
3. Punch a CR and LF.
4. Now punch the object code. After entering 32 bytes (64 hex characters), punch CR, LF. Repeat this for each 64 hex character line. (If tape reader is manually-pulled type, add about one foot of NUL's after each 256 byte block followed by the CR, LF. This will occupy about 4 1/3 feet of tape; longer lengths may be difficult to pull through and load properly.)
5. After end of object code, punch about one foot of trailer (NUL's).
6. If reader is manually-pulled type, and using black tape, a visual indicator of each block ending is convenient during loading. This can be added using white typing correction fluid, such as 'Liquid Paper'. This simplifies recognizing the trailer gaps between blocks.

## TABLE II

### ASCII - HEX CONVERSION TABLE

| ASCII - HEX | | | | ADDER | | |
|---|---|---|---|---|---|---|
| Character | Binary | | Hex | Inputs | | Output Sum = Hex value |
| | | | | A | B | |
| 0 | 0011 | 0000 | 30 | 0000 | 0000 | 0000 |
| 1 | ‖ | 0001 | 31 | 0001 | ‖ | 0001 |
| 2 | ‖ | 0010 | 32 | 0010 | ‖ | 0010 |
| 3 | ‖ | 0011 | 33 | 0011 | ‖ | 0011 |
| 4 | ‖ | 0100 | 34 | 0100 | ‖ | 0100 |
| 5 | ‖ | 0101 | 35 | 0101 | ‖ | 0101 |
| 6 | ‖ | 0110 | 36 | 0110 | ‖ | 0110 |
| 7 | ‖ | 0111 | 37 | 0111 | ‖ | 0111 |
| 8 | ‖ | 1000 | 38 | 1000 | ‖ | 1000 |
| 9 | 0011 | 1001 | 39 | 1001 | 0000 | 1001 |
| A | 0100 | 0001 | 41 | 0001 | 1001 | 1010 |
| B | ‖ | 0010 | 42 | 0010 | ‖ | 1010 |
| C | ‖ | 0011 | 43 | 0011 | ‖ | 1100 |
| D | ‖ | 0100 | 44 | 0100 | ‖ | 1101 |
| E | ‖ | 0101 | 45 | 0101 | ‖ | 1110 |
| F | 0100 | 0110 | 46 | 0110 | 1001 | 1111 |

NOTE: Observe that the hex code for characters 0 - 9 are identical to the 4 LSB's of ASCII codes 30 -39: further note that the hex code for A - F can be represented by nine (1001) plus the 4 LSB's of 41 - 46. Thus, using bit 7 to add zero (for bit 7 = 0) or nine for bit 7 = 1).

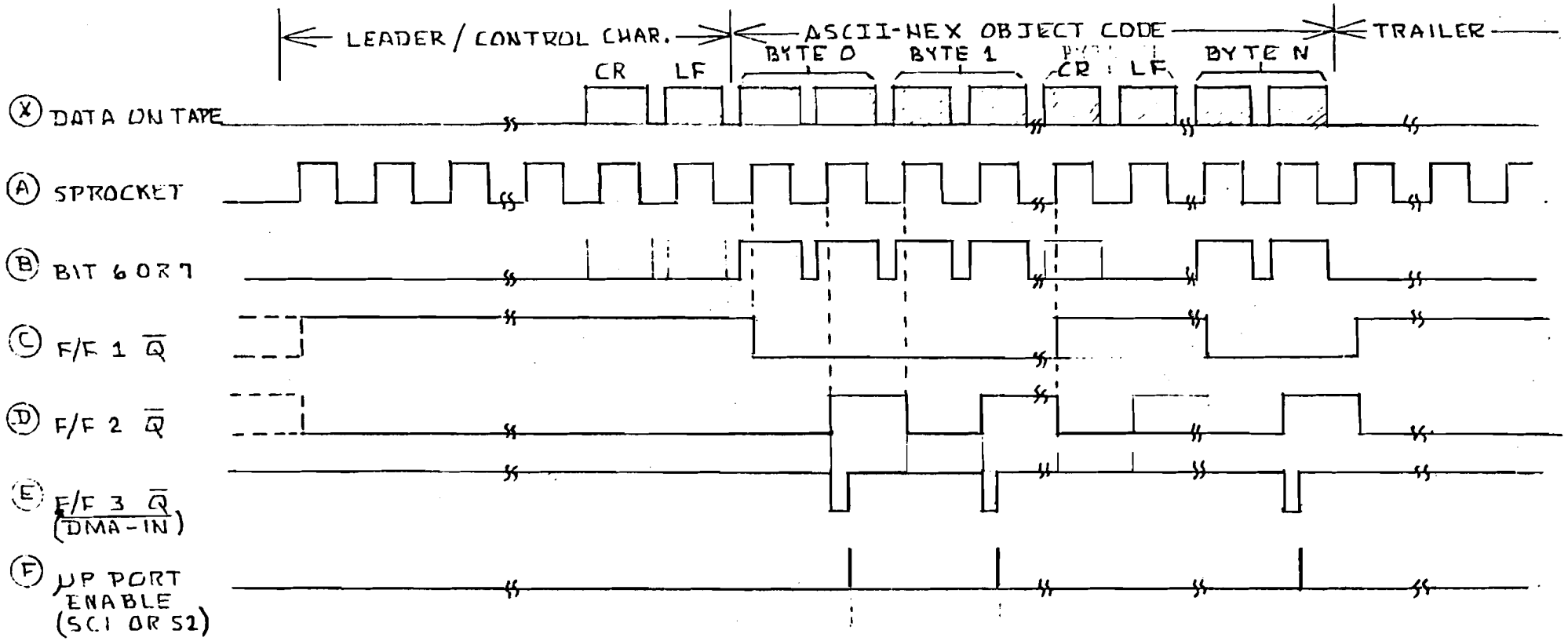
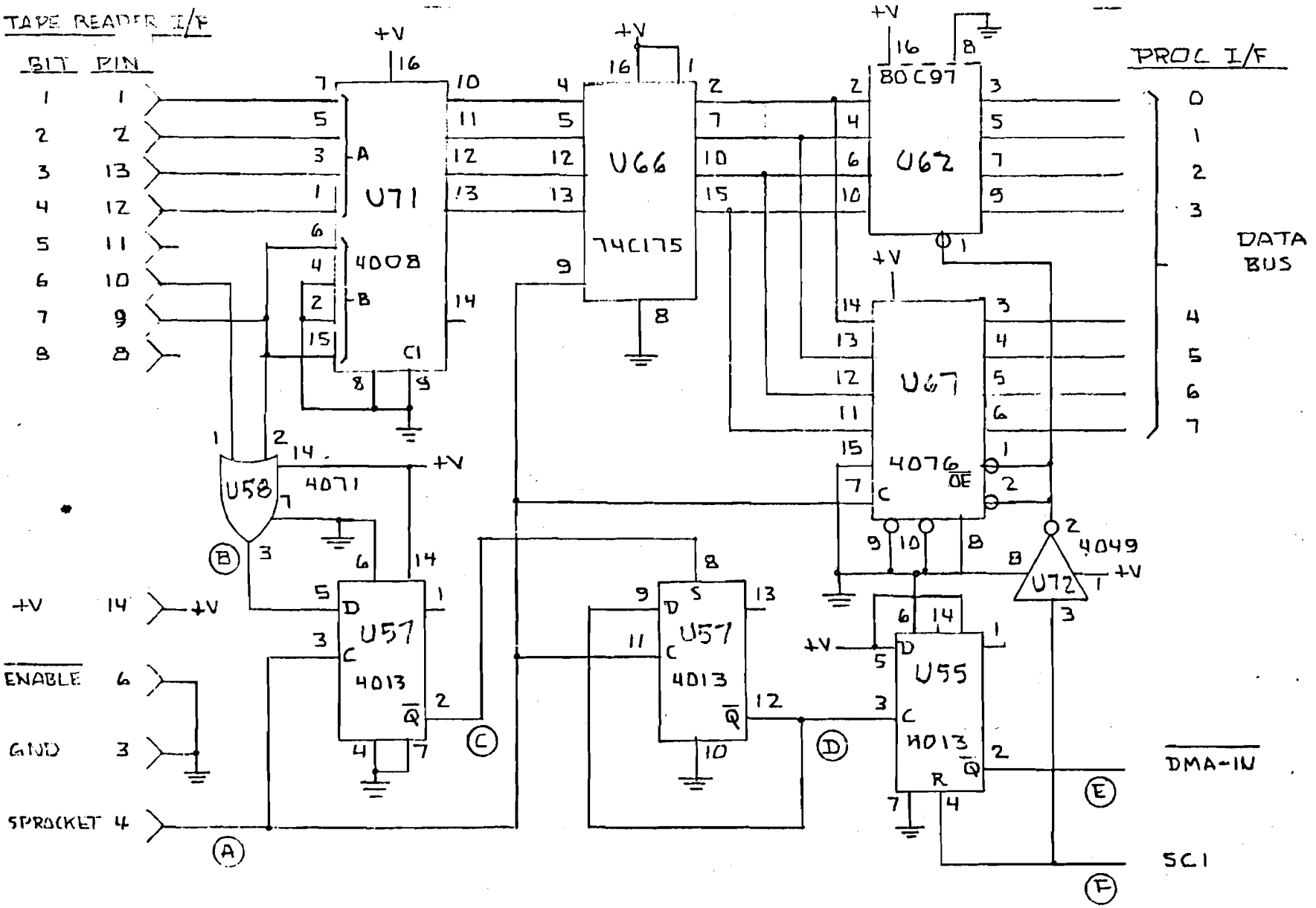





Fig. 2 HARDWARE PAPER TAPE LOADER – TIMING 29 JAN 78




(6)

Fig 3, HARDWARE TAPE LOADER-

24 JAN 78

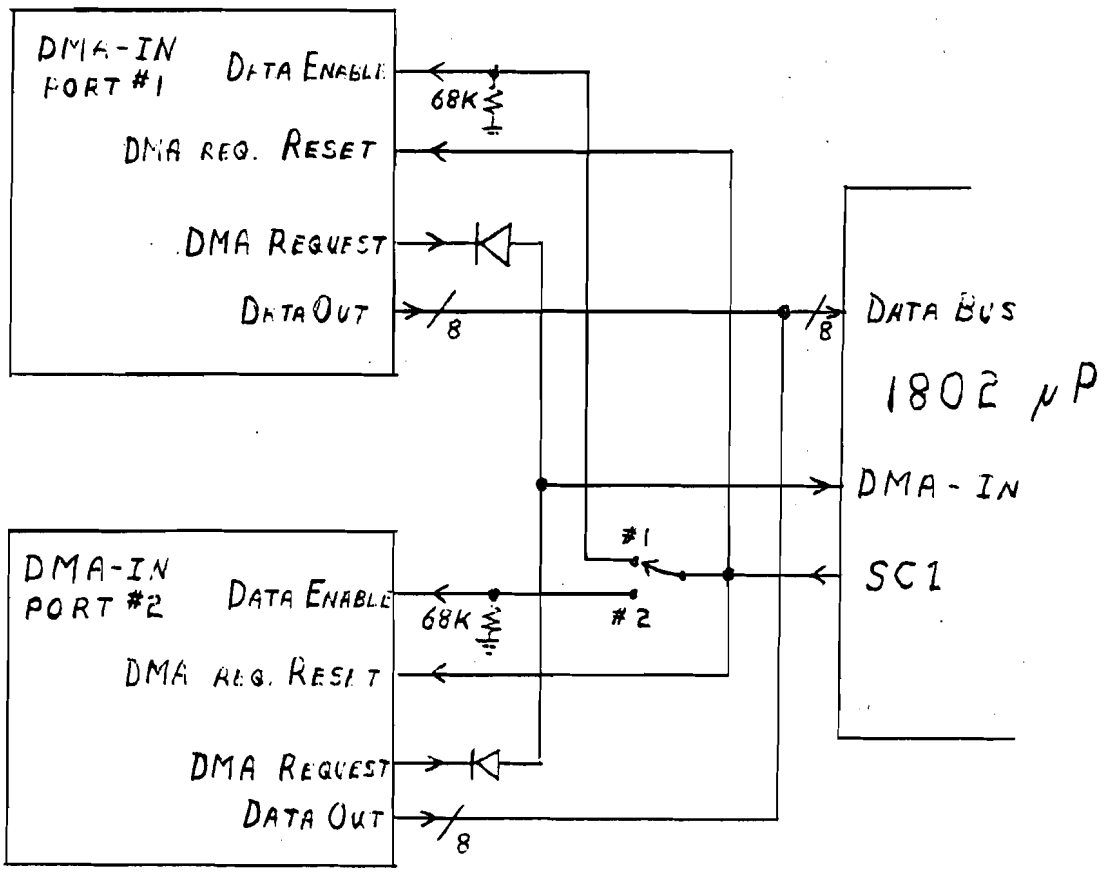



FIGURE 4

Editor's Notes:

1. The paper tape reader mentioned in the article is a Model TPR-1 optical paper tape reader, available assembled and tested for U.S. $32.50 from RAECO, Box 14, Readville, Mass. 02137 USA. Write for more details, or see one of RAECO's ads in most of the hobbyist computer journals.

2. The TEC-1802 keyboard operates in a DMA mode for program entry. Anyone care to write an article showing how they interfaced this paper tape reader to a TEC-1802?

## IS YOUR MICROCOMPUTER S-100 COMPATIBLE

Wm. Lee Pfefferman

I'm sure that by now you all are aware that we COSMAC users are minority members in the world of microcomputing. Being minority members, we are in a sense cut-off from the mainstream of software that is readily available to the majority. Unfortunately, many of us are members of still another minority; we operate microcomputers that are not compatable with the S-100 bus and this serves to isolate us from the mainstream of hardware that is available to the majority.

For me, S-100 compatibility will make it easier for me to accomplish the goals that I have set for my computer system and I am currently in the process of making my COSMAC-based microcomputer compatible with the S-100 bus. Having an S-100 compatible computer will enable me to do several things that I wasn't able to do previously.

1. It will be easy for me to EXPAND on the existing features of my computer, memory or floppy disk controllers can be added almost at will.
2. I'll have the ability to CHANGE the existing features of my computer, in whole or in part. Changing to a different microprocessor may not be as easy as replacing a single PC board, but I don't think it will take too much effort.
3. I can INCREASE the CAPABILITIES of my computer by adding boards that perform a wide variety of functions. S-100 boards now exist that will control floppy disks, synthesize speech, and analyze the operation of a computer.
4. I can IMPROVE on the existing parts of my computer by referencing the large amount of literature that has been written for S-100 compatible systems.

By changing to the S-100 bus I will also benefit economically. I will have the ability to select PC boards from many competing vendors, this will assure me of being able to purchase top quality boards at low prices. In addition, I will be able to sell and trade my boards with many more people than I could before. Besides the obvious economic benefits, I hope to use this capability to keep the structure of my system somewhat dymamic in nature.

As you can see, in my case, the advantages to having an S-100 compatible computer are several and I can hardly wait to look through the advertisements in the next issue of BYTE.

## LOGIC TESTER

Tom Jones

I designed this tester from available parts to aid in TTL trouble-shooting some time before I got into CMOS. I added the CD4050 to allow CD1802 design work. It has helped to resolve some mainframe problems where my Tektronix 453 could not help me--due mostly to the TRAP or MEMORY mode.

To use, set S2 to pulse and connect either pin 1 (for TTL) or pin 4 (for CMOS) to the test point and observe the 7-segment display for a "1" or "0" or "P". Also observe the decimal point for a "pulse stretched" indication. A square wave will give a "P", but very narrow pulses far apart will only keep the decimal point on. The decimal can now be used to watch for positive or negative pulses.

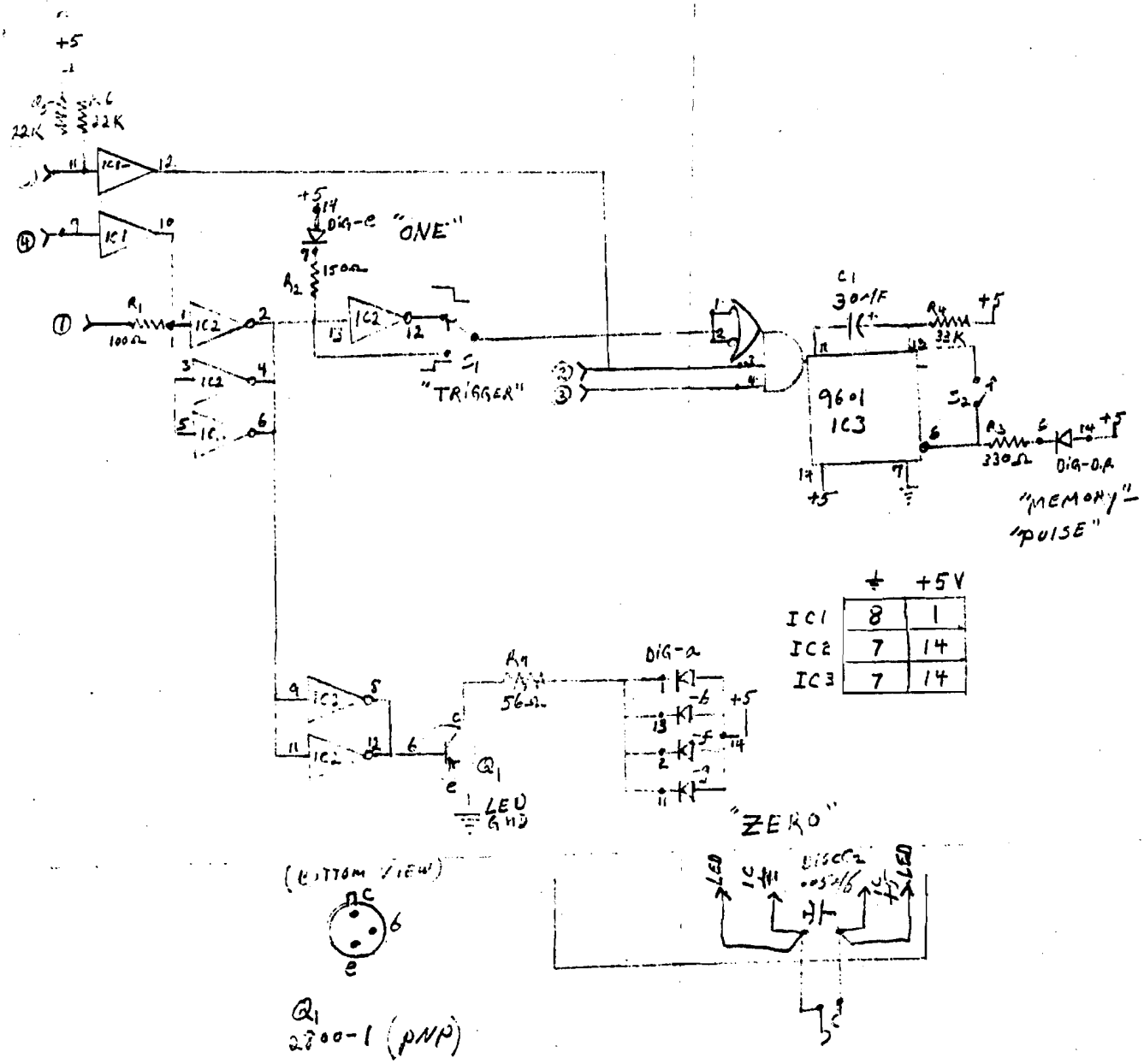
If a solid "1" level is observed, you can test for a random negative pulse for a long period by setting the "⊓" on S1 and setting S2 to "TRAP" mode, and go to lunch. If the solid high (perhaps 5 volts) ever goes low, the decimal will set. In fact, if the tester is being powered from the supply, the trap will set even when the probes are not connected if power fails.

If a solid "0" were monitored you could of course check for positive pulses by setting S1 to "⊔" and S2 to "TRAP" as before.

In many computer problems, it is necessary to gate the input with another function (such as TPB, MRD, or DATA VALID). Connect input 2 (TTL) or 5 (CMOS) to this function to make sure you are observing the signal only during the period of time you are interested in. Choosing this function can be quite a bit of an art, and is not limited to data strobes. An auxiliary flip flop that is set by one function and reset by another function can provide this input on 2 or 5, thus allowing you to choose whatever timing window your imagination can create. I've done it. The other leg of the gate on IC3 is brought out for TTL but not for CMOS on mine, I may bring it out for CMOS if I ever need it.

My unit is housed in a small wooden box (a doctor gave me several, I have no idea what he receives in them). I wired every thing on perfboard using a wiring pencil and sockets. The transistor is house numbered, but any good PNP will work. The I/O jack is half of a 14 pin wire wrap socket, and the plug is the other half with the pins cut short and carefully rounded with a file. I simply push the wires in the socket holes and seal them in with black rubber compound such as the G.E. stuff sold for sealing car windows. Long leads for inputs and power and very flexible wire will avoid much frustration in the future.

Some features I would like to add to any unit I build again: battery power, to avoid using the system power. The system power may be the problem, and anyway the hook up wires are a bother, and will ruin the unit if connected backward. I would like to add a feature that shows an "F" if the input voltage is not a valid "1" or "0", also. The present unit shows a floated pin as a "1" due to the pull-ups. I think some cheap diodes would make the "e" and "f" segments both light for a "1" level, instead of only the "e" as presently. I leave these to anyone else interested in his own tester, and hope my experiences using mine will suggest better ideas to you.




| | ⏚ | +5V |
|---|---|---|
| IC1 | 8 | 1 |
| IC2 | 7 | 14 |
| IC3 | 7 | 14 |

13 b
1 a
2 f
7 c
6 DP
11 g
14 - +5V

7-SEG. DISPLAY: HP 5082-7750

## LETTERS TO THE EDITOR

Dear Tom:

I am currently working on an interface between the 1802 and a National Semiconductor's 57109 number oriented processor. If the hardware result of this could be software coupled to a basic interpreter such as the tiny basic available from the ITTY BITTY COMPUTER CO., the result would be a pretty tasty system with the number crunching done by the peripheral processor. (How does a 2K interpreter capable of floating point or scientific notation and functions such as all trig functions and their inverse, square, square root, log, antilog, LN, exponentials, inverse, degrees-to-radians, radians-to-degrees, etc. grab you?)

I hope to have some notes available soon if anyone is interested. If anyone else has any ideas, I'd like to hear from them.

Yours truly, Dave Hayward, 6640 Fielding, Apt. #12, Montreal. Quebec, H4V 1N3

Dear Mr. Crawford: February 6th, 1978.

Received first newsletter last month, keep up the good work!

Just a few comments on my system and the problems I have encountered. I am running now an ELF II (no doubt you have heard of it!) and am running it with 1024 bytes of memory. I put this together with 2102's bought from S.D. Sales.(it's a good value, 11 to 12 dollars for 1K.)

I received Wayne Bowdish's cross-assembler too, although I fear it will be a struggle getting the ------ to run on our PDP-8/E. (has anybody done it yet?)

I have done little work with the computer lately but have succeeded in getting a few of my ideas working; i.e. hexadecimal counter on the television screen, "t.v. typewriter" - four lines of eight characters! and a memory tester. (sorry but I've no time to write them up in good)

If would print this request for me as soon as possible? I'd like to know if anybody has considered a game of Tic-Tac-Toe as a machine language project-please send me your ideas.

Has anybody considered a light-pen for the CDP1861 graphics IC chip? Please send software that you have implemented.

I have to go now so - keep up the good work on the newsletter and keep well.

Yours truly, D'Arcy Roberts, 660 Laurier Blvd., Brockville, Ontario. K6V 5X8

Dear Tom: Jan. 14, 1978.

I am interested in the 1802 User's Group. I just purchased a RCA VIP, and am very pleased with it. I think it has the potential of the KIM-1, and am considering starting a VIP Newsletter. However, I don't wish to duplicate your efforts. Do you plan on doing a regular newsletter? Are you interested in the VIP? (Ed. Note: I understand RCA is starting a VIP User's Group. TC)

Some quick notes on the VIP: Kit assembly went well. RCA estimates 3 hours for assembly; it took me 4 hours (I'm a fairly experienced kit builder), plus another 4 hours of wiring and debugging. My 5 volt power supply had an intermittent short which reduced the voltage to about 2 volts; after cutting the plastic

sealed assembly open, I discovered a diode lead dangling next to the heat sink. I bent it into a safer position, and have had no further problems.

I'm using the "Waterloo" RF modulator (BYTE, Jan '78), and find it works well, although it interfers with other channels (even when not directly connected) if the RF lead is 6 feet long. I like to lie in bed and play with the VIP, so I needed a long connection to my TV. By putting the modulator near the TV and running a long video line instead, the interferance was eliminated. I used the VIP's 5 volt supply to power the modulator.

Documentation provided with the kit is pretty good. The logic description, trouble-shooting guide, and test programs are excellent; the ROM operating system guide is very good (the operating system itself is adequate); the CHIP-8 interpreter guide is only fair, however, and there is no description of how each of the 20 video games work.

The video games themselves are pretty good, but don't expect quality. "Wipe Out" and "Kaleidoscope" are especially fun (I've only tried three so far). My first project will be the game of "Life".

The CHIP-8 interpreter provides a real high level (although hexadecimal) language for controlling the display, reading the hex keyboard, etc. It is easy to learn, and certainly one of the best parts of the system. The interpreter itself, by the way, takes 512 bytes of RAM, and the first thing the VIP user must do is enter the interpreter code (in hex) and save it on tape.

Hardware expansion is easy; adding a parallel I/O port and 2K more of RAM is just a matter of plugging in chips. The parallel I/O port could use more handshaking lines, however (there is only one output handshaking line, which is also used on the board for other things).

In summary, it's a fine system, and provides lots of room for both hardware and software experimentation. It's the perfect system for the computer hobbyist just getting started, or waiting for "the ultimate system".

Sincerely, Bob Wallace, CoMind Design, PO Box 5415, Seattle, Wa 98105

Dear Tom Crawford:

Thank you for your kindness and for the time you spent to get me all the information on A.C.E.

I'm really glad to hear, that there is so many of you who work on 1802 systems! Here in Vancouver only a few of us have 1802 CPUs - mostly 8080 - Z80 or 6800 the favourite one. We have a computer club also that hold meetings on the first Wednesday of every month and also publish a newsletter monthly. In the future I'm going to send you a copy of our newsletters. On our next club meeting I'm going to display your letter and I'm sure will have some very favourable response about information exchange between our two clubs.

Please say HELLO to A.C.E. members from our Vancouver club!

Thank you again, will be in touch soon!

Sincerelly, Veslot Gellerthegyi, No. 305-1315 Broughton St., Vancouver, B.C. V6G 2B6

LETTER TO THE EDITOR CONT'D

Dear Tom:

I read with interest the fact that there is a 1802 club formed in the Toronto area, from a copy of Byte. I appear to be the only one in the Halifax, Nova Scotia area using the u-P at least from the club meeting here that I have attended. I am using a P.E. COSMAC ELF board with outboard 4K memory. I/O consists of a ASCII Keyboard, paper tape reader and a model 19 Teletype, of course.

I have been programming in machine language up until this point in time as I am not proficient enough in software to prepare an assembler. While I expect that you are aware of this already, I note that Infinite Inc. 1924 Waverley Place, Florida have Tiny BASIC as well as a few other small programs available. I am ordering this and if you wish can report on any success that I have with it. I am enclosing a photocopy of their available programs. It appears that only the BASIC related programs are of much value as the others can be done quite easily by most anyone who has done anything with the chip.

About the only thing that I have done that might be of some use to your club might be my program which is a print using Baudot TTY subroutine. This will take ASCII data, either immediate from the keyboard, or a block stored in memory, code convert it to Baudot and serialize it (as a UART would). It outputs on the Q line and is optoisolated from the selector magnets. As it takes no I/O port or hardware other than an optoisolator and a few transistors and resistors, it is a very cheap way to get the 1802 talking especially if you can get an old model 15 or 19 free as I did.

I would be interested in hearing from you if you have any software available. I would be interested in an assembler and possibly BASIC if the Infinite software doesn't map too well with my setup. Also I would be interested in any work that is being done in either 1702 or 2708 EPROMS as I would like to have some form of monitor and/or paper tape loader in ROM as I am getting tired of starting from scratch every time I shut down to do something to the Hardware....

Thanking you in advance, I remain
Yours very truly, Brian Millier, Box 34 Site 12A, R.R.3
Armdale, N.S.

Mr. Doerwald:

Thank you for acquainting me with your group, I was a bit surprised at the size and activity of the 1802 group you describe, I had thought we were odd birds. I have heard of one group in Texas. My application fee is enclosed.

I usually like to contribute what I can to any club I join, but at the present, I have little to offer. I am only just getting my home-brew 1802 on the air. I started the design a year ago, but time and money--well, we all know the story. I can send one proven design, my logic probe, which I have used on CMOS at home and TTL at work on the job. (I am a computer field engineer representing Honeywell Information Systems.)

What I have is an 11" by 5" perf board half full of wire wrap, a TTL hex keyboard, and a paper tape reader. On the board are a CPU, 2K of 21L02, a parallel port and a serial port, and an unfinished UART based cassette interface. (I don't intend to finish it, but to use the Netronics I/O and their monitor in RAM.) I have already decoded on the board all the N lines and 32 mapped memory I/O pulses.

On the drawing board still are a CMOS (mostly) TVT, a led 8 digit panel and ASCII keyboard combo, and a simple A/D-D/A port aimed at a DVM program. First on the list, however is a new CMOS HEX keyboard utilizing the 74C922 LSI chip and the best Keypad I can find. My keyboard is a handicap.

Some design thoughts:

I suggest that the use of simple I/O driver routines located in ROM monitors was a shabby attempt by many hardware manufacturers to nail users into thier systems and a practice not to be blindly followed by the users themselves. I believe the 1802 group is still not much influenced by manufacturers. The only routine I plan to put in ROM at present is a bootstrap loader geared for K.C. standard cassette format. (The UART driving DMA is too complicated). My tapes will all have a short BRT record at the beginning which will be a basic loader for the rest of the tape. This followed by one or the other monitor. (The Basic Loader will read the starting location of each record on tape and set the X reg to that.) The only remaining task is the I/O port assignment, and since each peripheral has an subroutine in the Monitor, it is easy to give the user a chance to enter each port in turn and plug them into the routines during startup. This scheme is not only to help others use my stuff, but to help me--- I change my system constantly also. But any 1802 that can boot in that first record could probably run the program quickly.

My other dream is a monitor subroutine that replaces each jump or call in a program and makes possible dynamically relocatable code on an 1802. I lean towards the standard 44 pin bus for system design, but the idea of standards in the experimenter world is doomed in my opinion, so the best scheme is to be as general and flexible as can be. The tape format I am designing is an example.

Thank you for your letter, I hope this will introduce me, and I shall look forward to recieving future issues of IPSO FACTO.

Yours truly, Tom Jones

Dear Sirs:

...I have considerable electronics experience as concerns hardware but am strictly a beginner as concerns software.

I have built a small 1802 basic system as described in Popular Electronics. It is mounted in a Hammond case and I have been able to program it with the help of the above magazine. I have adapted a program from P.E. to turn the Q LED and a speaker on and off under control of a program which I use with Cibachrome colour processing. Even using a CD4047 as a clock, this micro makes a super timer (variable Q time intervals).

However, I plan to build another for general purpose usage and my programming skills will have to be greatly increased.
Looking forward to your newsletters.
Yours very truly, Brian Rusk, 1853 Arizona Ave., Ottawa KiH 6Z5

Dear Tom:

...I will soon receive the COSMAC VIP version of the 1802 based systems and eagerly look forward to a long and hopefully mutually beneficial relationship with A.C.E. For the time being, I'll only have the VIP, a monitor and a cassette machine as I am doing a product review for QST, the journal of the American Radio Relay League for whom I work. When that is done, I intend to expand the memory, somehow work up a full ASCII interface and go from there.
In some respects, I don't share the all too common problem of trying to figure out what to do with a microprocessor system--as a ham I have all too many practical applications. Are there other hams in the club from the Ontario, i.e. generally, "local" area? If I can be of any amateur radio service to them at the ARRL, please tell them to feel free to call me at any time. My job there is OSCAR (Orbiting Satellite Carrying Amateur Radio) Education Program Manager...
Thanks for your time and cooperation. I've got a lot to learn and hopefully, in the near future, I'll have a lot to offer.
Sincerely, Steve Place, 271 Williamstown Court, Newington, CT 06111

Dear Tom:

...I own a ELF II and am adding 4K of RAM and the RCA Binary Arithmetic Subroutines to the Standard Call and Return Technique in a 2708. My next move is to add the Number Cruncher from National along with decoded hex display of upper and lower address bytes.
Some information on parts availability for your readers NSN373 dual 7 segment display for the Electronics Experimenters Handbook 1978 - 1802 display is available from Digi-Key Corp. P.O. Box 677, Thief River Falls MN 56701 for $2.20 each. 9368 decoder/driver for the above is available from Active Electronic Sales, 44 Faskan Drive, Unit 25, Rexdale, Ontario for a decent price. 86 pin wire wrap connectors that fit the ELF II bus are available from Advent Electronics, 24260 Indoplex Circle, Farmington, Michigan 48024. Their code number is 172-0043-005. They can be soldered to the existing bus and wire wrapped for user's needs.
It is great to hear about other 1802 projects and ideas. That way I don't work in a vacuum. Thank you. P.S. I would also like to hear from any other 1802 users in the Iowa, Missouri, Illinois area.
Yours Sincerely, Alan Bwines, M.I.U. Extension Dept., Fairfield Iowa, 52556

LETTERS TO THE EDITOR CONT'D

Dear Mr. Crawford:

Following article extracted from Interface Age magazine might be interesting to our club members.

A byte of music by C.G. Smith in the Nov. issue of Interface Age teaches how to play music with RCA 1802 Cosmac microcomputer. The main program written in machine code for RCA 1802 and 3 J.S.Bach minuets are included in the article. How it works and how to convert the music tunes to music bytes are discussed in the article. The main program is 89 bytes long, so if you have only the basic RCA 1802 with 256 bytes RAM, you can already entertain your family with real music from the micro. How is the interface -- ? --- The interface required is only a piece of wire. Connect one end of the wire to the LED attached to the Q output and the other end to the antenna of the radio, place near an AM and tune the radio. Some characteristics of the program are:

1. The program works on the RCA 1802 Cosmac microprocessor.
2. Only one memory byte is required to give both note frequency and duration parameter.
3. The music is started over again if the end of the song is sensed.
4. It is easy to convert real music into music coded form.
5. It is possible to play rest by placing Zeroes in the 'note bits'.

ENJOY YOUR MUSIC AND YOUR ---- MICRO !!!!!

Yours very truly, Mrs. Sianhoei Lie

A SINGLE CYCLE CIRCUIT FOR THE 1802 Wm. Lee Pfefferman

Many of you, I'm sure, are aware of the technique of single-stepping a microprocessor to monitor the execution of a program. The 1802 has the ability to be single-stepped and a circuit for single-stepping can be found on P.72 of the COSMAC User Manual.

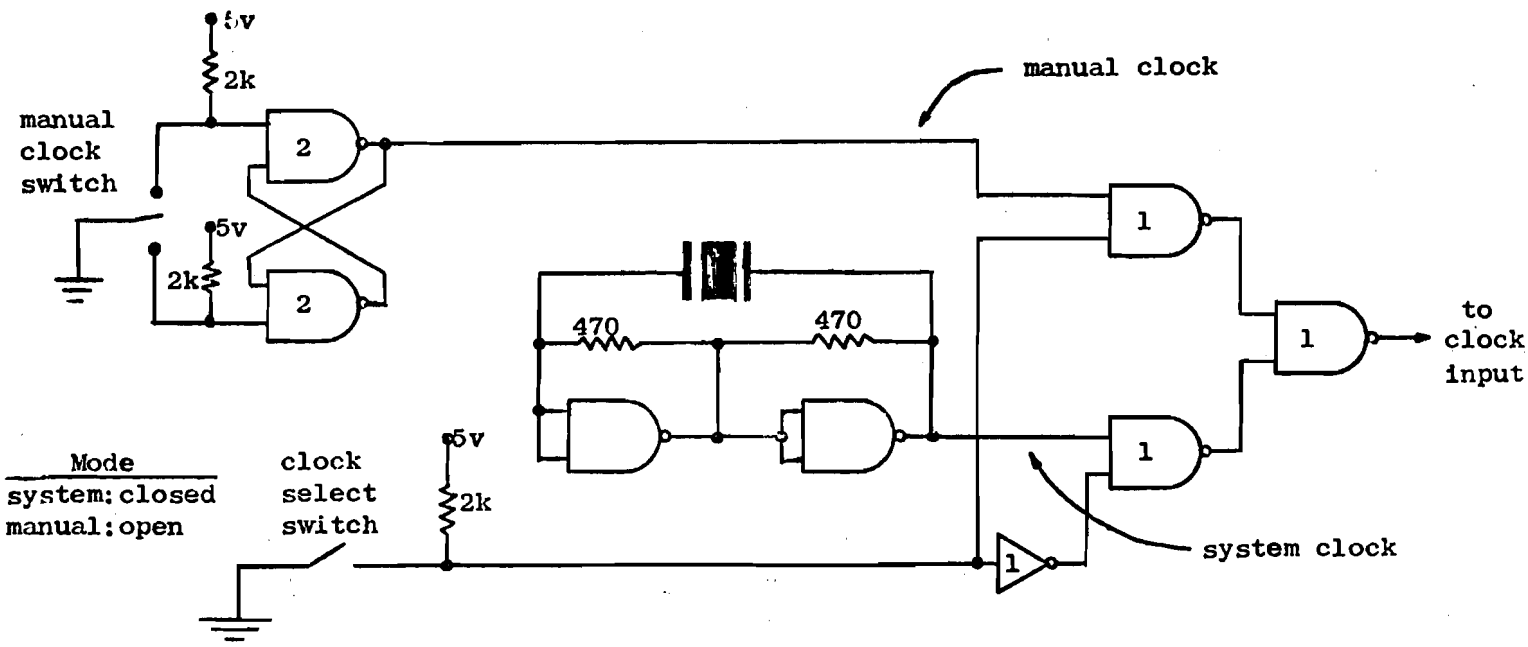
Unlike many other microprocessors, the 1802 has no minimum clock frequency. For 1802CD users, like myself, this means that our COSMACs can operate with clock input frequencies from DC to 3.2 MHz. I have made use of this capability to construct a single cycle circuit. This circuit facilitates the debugging of both hardware and software.

The circuit operates by setting a select switch to one of two positions.

1. Setting the select switch in system mode connects the system clock to the clock input on the 1802. This allows the COSMAC to operate normally.
2. Setting the select switch to manual mode connects a SPDT switch to the 1802's clock input. Using this switch to generate clock pulses allows the user to inspect all information on the COSMAC data and address busses, eg. input and output data, op-codes, register and memory contents. Please make sure to note that this scheme assumes that the contents of the busses can be displayed without being latched.

## A SINGLE CYCLE CIRCUIT FOR THE 1802 CONT'D

The circuit that I use is shown below.




The heart of the circuit is the 2:1 multiplexor, constructed from the gates labeled #1. The multiplexor is used to pass either the manual clock or the system clock to the COSMAC's clock input, depending on the position of the clock select switch.

I use the S-R flip-flop, gates labeled #2, to de-bounce the manual clock switch.

Finally, the rest of the gates are used to construct a standard TTL oscillator circuit.

## A FINE RESOLUTION AUDIO OSCILLATOR PROGRAM

R.G. Edwards

Some people may want to realize a fine resolution for audio tone generation using a 1 MHz crystal. The following subroutine may be useful.

```
TM2         B4  *       Debounce  EF4
            SEP 3       Return to calling Program via R3
TONEGEN     REQ         Entry Point for Subroutine
            B4  TM2     Quit on EF4 Interrupt
TG1         LDN 9       R9 is Pointer to Delay Table
            SR          LSB to DF
            LSND        If LSB is a Zero-5 Word Times
            LSD         If LSB is a One-6 Word Times
            NOP         (3 cycle NOP)
            SEX 9       (2 cycle NOP)
            SR          BIT 1 to DF
            BND TG2
            SEX 9       If BIT 1 is 1-Add 2 Word Times
```

```
TG2     SMI 01      4 Word Time Delay Loop
        BNZ TG2     Using R2
        BQ  TONEGEN
        SEQ         Set Q and Generate the other Half of
        BR  TG1     the Squarewave
```

Notice the routine puts out a symmetrical square wave.

The subroutine produces tones with a full cycle resolution of 16$\mu$s with a 1 MHz crystal. The formulae for the tones are

$$\text{Delay} = \frac{31250}{f} - 19 \qquad f = \frac{31250}{D+19}$$

Example:

Delay constant for $f = 440$

$$\text{Delay} = \frac{31250}{440} - 19 = 52.022727 \approx 34_{HEX}$$

Frequency for delay = 52

$$f = \frac{31250}{52+19} = 440.14084$$

Delays for a true tempered scale are:

| | | | |
|---|---|---|---|
| A | 7B | E | 4C |
| A# | 73 | F | 46 |
| B | 6C | F# | 41 |
| C | 64 | G | 3D |
| C# | 5E | G# | 38 |
| D | 57 | A | 34 |
| D# | 51 | | |

## CERTIFYING AUDIO TAPE FOR DIGITAL USE

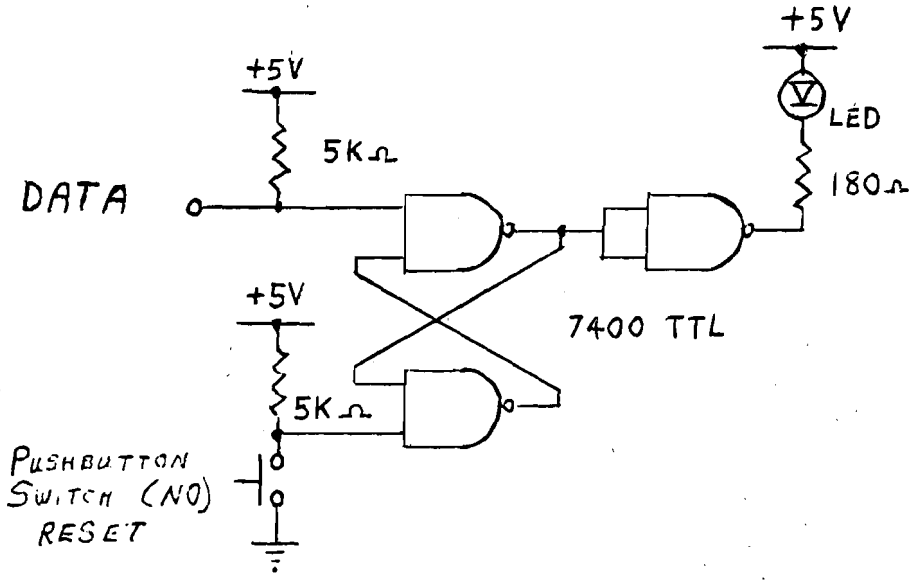
For best results a tape cassette should be tested before use to determine if it contains flaws which will create errors. Computer grade tape is subjected to a series of tests which 'certify' its freedom from such error producing flaws. Since 'Certified' cassettes sell for two to four times the price of high quality audio cassettes, you will probably prefer to test the quality audio cassettes yourself. The test to be described is not as thorough as the computer grade certification procedure but it is more than adequate for the hobbyist.

The procedure is simply to record a continuous signal on tape then play back at reduced level and let the cassette interface (with some additional circuitry) watch for loss of signal. If the tape passes the test at reduced play-back level it is almost certain to be adequate under normal level conditions.

Procedure:

1. Record a continuous 2400 Hz tone at normal operating level on the cassette. This can be done by connecting the tape recorder to the Kansas City Standard Cassette Interface since the Interface generates a 2400 Hz signal when idle.

2. Reduce the playback signal level to half of the normal level.
3. Connect the cassette player to the K.C. Standard Cassette Interface and run the tape. If the Data output is connected to the following circuit, any flaws will cause the LED to illuminate until reset with the pushbutton switch.




## A COIN TOSS PROGRAM

Robert C. Taubert

I have a COSMAC ELF II, purchased from Netronics R&D. I have included a brochure and schematic of the ELF II. I have made no modifications or additions to my ELF though I have some planned for the future. I would like to add at least 4K of memory, an ASCII keyboard, a monitor, a cassette interface, and some form of AD-DA board. I would like to begin running Tiny Basic as soon as possible with hopes of advancing to 4K or larger in the future.

My first attempt at writing a machine language program was somewhat frustrating and exciting. I wanted a program that would simulate 15 flips of a coin and display the total number of heads on the left LED and the tails on the right LED. My next project will be to add a delay sub-routine to allow me to increase the number of tails.

I have enclosed a copy of the initial coin toss program, which I wrote, and the random number generater, which I obtained from Ross Wirth's newsletter. The program is simple but does use many of the 1802's instructions.

## COIN TOSS

```
00    F8  07  A3     *Load R(3).0 with address of "MAIN" program
03    F8  51  A4     *Load R(4).0 with address of "RND" subroutine
06    D3             *Set P to R(3).0     "MAIN" program counter
07    F8  43  AB     *Load address of "HEADS" in R(B).0
0A    F8  00  5B      Zero "HEADS"
0D    F8  44  AC     *Load address of "TAILS" in R(C).0
10    F8  00  5C      Zero "TAILS"
13    F8  45  AD     *Load address of "TOSS" in R(D).0
16    F8  00  5D      Zero "TOSS"
19    F8  46  AE     *Load address of "RESULT" in R(E).0
1C    F8  00  5E      Zero "RESULT"
1F    D4             *Call subroutine "RND"
20    0D  FC  01     *Increment "TOSS"
23    5D
24    0A  FA  01     *M(R(A).0) AND 01 to D
27    32  2F         *IF D=0   GOTO  2F
29    0C  FC  01     *Increment "TAILS"
2C    5C
2D    30  33         *GOTO  33
2F    0B  FC  01     *Increment "HEADS"
32    5B
33    0D  FB  0F     *"TOSS" XOR 0F   (0F = D.15 = Max. # of Flips)
36    3A  1F         *IF D ≠ 0  GOTO 1F
38    0B  FE  FE     *Shift "HEADS" 4-Bits left
3B    FE  FE
3D    EC  F3  5E     *"HEADS" XOR "TAILS" & store in M(R(E).0)
40    EE  64  00     *Display "RESULTS" & STOP
43    HEADS  TAILS
45    TOSS   RESULTS
```

## REGISTER ASSIGNMENT

| | | |
|---|---|---|
| R(3) | MAIN | Program Counter |
| R(4) | RND | Program Counter |
| R(8) | RND | D* |
| R(9) | RND | Count |
| R(A) | RND & MAIN | Seed & Rnd # |
| R(B) | MAIN | Heads |
| R(C) | MAIN | Tails |
| R(D) | MAIN | Toss |
| R(E) | MAIN | Result |

A COIN TOSS PROGRAM CONT'D

RANDOM NUMBER GENERATOR

```
50    D3                 *EXIT
51    F8  7A  A9         *Load R(9).0 with address of "COUNT"
54    F8  7B  A8         *Load R(8).0  with address of "D*"
57    F8  7C  AA         *Load R(A).0 with address of "SEED"
5A    F8  00  59         *Zero "COUNT"
5D    0A                 *Get "SEED"
5E    FA  8E  58         *AND "SEED" with 8E & store in "D*"
61    32  73             *IF D=0   GOTO 73
63    09                 *"COUNT" to D
64    FB  FF             *D XOR FF to D
66    FA  01  59         *D AND 01 to D to "COUNT"
69    08  FB  FF         *D  XOR FF to D
6C    FC  01             *D + 01 to D
6E    E8  F2  F3         *D* AND D to D, D* XOR D to D
71    30  60             *GOTO 60
73    0A  FE             *Shift "SEED" 1-Bit left
75    E9  F4  5A         *"COUNT" + D to D
78    30  50             *GOTO  "EXIT"
7A    COUNT
7B    D*
7C    SEED
```

MAGNETIC TAPE DATA RECORDING — Ken Smith

Recording digital data on magnetic tape or disc is still (and always has been) the most cost effective for mass storage, in both professional and hobbyist fields. Although a tape recorder is very common and is conceptually simple, things get involved and complicated when it comes to instrumentation and digital data recording. There are several factors which determine the performance of a magnetic data recording system.

1. Data density (bits per inch)
2. Data transfer rate to and from tape
3. Access time to a particular part of the tape
4. Method of recording eg. frequency shift, RZ, NRZ, Phase encoding, Kansas City, etc.
5. Data format: preambles, records, inter-record gaps, etc.
6. Coding: this is to combat errors eg. parity, checksums, cyclic codes
7. Error rate.

Most things boil down to one thing; what about errors? This is the annoying aspect of data recording. There are several things which cause errors.

1. Tape dropouts due to bad tape
2. Dirty heads
3. Dust
4. Dirty or poor tape transport system
5. A cassette which binds and jams or causes too much friction.

The first solution is obvious, use your expensive stereo reel-to-reel deck. But almost all people use an inexpensive cassette recorder. Use good tape to begin with. Cheap bargain tapes are tempting but are unpredictable for dropouts, etc. Maxell UDXL C-60 is one of the best for computer applications. Sony is also good.

Don't forget that in a cassette, half of the transport is the cassette itself. A bad cassette will make a squealing noise when rewinding. As for the cassette recorder itself, make sure it does not 'thew-up" the tape. Look at the tape which has been played a few times. A bad recorder will put longitudinal lines, scratches on the tape or fray the edges.

Clean the heads, guides, capstan and pinch roller with alcohol. Record a tone and play it back. Does it sound "sick" or "watery"? Does the recorder have hum or noise from the motor at the output? There are several reasons why a recorder (a cheap one) won't record or load programs.

SATURATION RECORDING

Almost all commercial digital tape units use "saturation recording". This method saturates the tape material in one direction or the other, which is what binary data is; one state or the other. Recording this way is very simple to do. The record head current is switched from one polarity to the other. The current must be sufficient to saturate the tape, typically ½ to 5 mA. Since we want to switch the magnetic flux, a current source (not voltage) is desired. The head is usually fairly inductive which impedes a change in current. Fig. 1 shows a simple circuit for saturation recording. Saturation recording considerably reduces dropouts on playback and hence errors. The value of C and R will depend on head resistance and inductance. Adjust R for minimum current for good saturation, C for best current transient response. $S_1$ is always in the recorder for switching between record and playback. Disconnect the record amp.

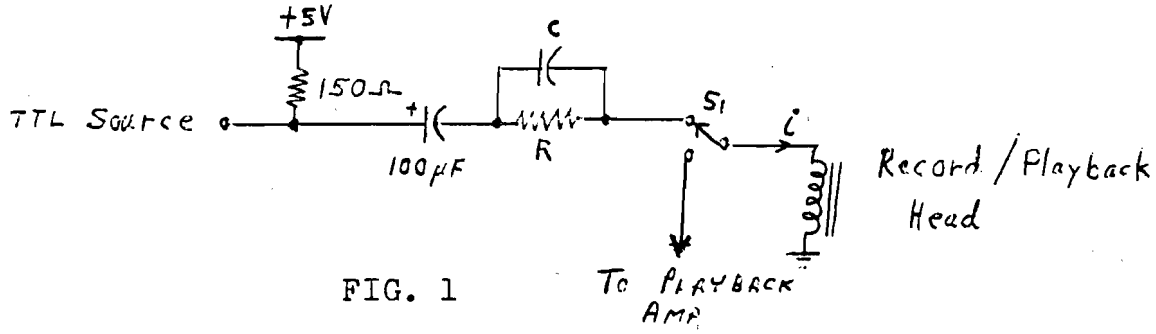



FIG. 1

PHASE ENCODING

Phase encoding, a favourite of mine, is commonly used commercially. This is a method by which data is recorded on tape, while Kansas City is another way. Fig. 2 shows how phase encoding works. The direction of change in the middle of the bit "cell" determines if the bit is 0 or 1. Phase encoding has many advantages. Like Kansas City, it can be A.C. coupled (no D.C.). On playback, the zero-crossings are detected, like Kansas City. But for a given maximum frequency response of the recorder, phase encoding gives 4 times the data rate that Kansas City will. 1200 baud phase encoding requires a frequency response only from 600-2400 Hz. 300 baud Kansas City requires 1200-2400 Hz.

## MAGNETIC TAPE DATA RECORDING CONT'D

In addition phase encoding is self-clocking; the code itself generates the data and the clock. It is also fairly insensitive to tape speed variations since the clock is regenerated by the code. Also, no start or stop bits are required. This is also true for Kansas City but not FSK.

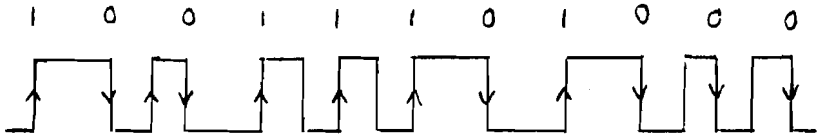



FIG. 2

Phase encoding is simply generated by exclusive-oring a symmetrical clock, at the baud rate, with direct data synchronous with the clock. Since phase encoding is simple to generate and recover (data and clock) simple software will replace the need for a UART.

Phase encoding has its problems. A cheap recorder with bad equalization may not properly reproduce phase encoding. We only worry about the zero-crossings. Also if the playback signal is inverted, the data gets inverted. You must determine if this happens with your recorder.

## A SIMPLE 25 IC - 2 TRANSISTOR CODE PRACTISE OSCILLATOR

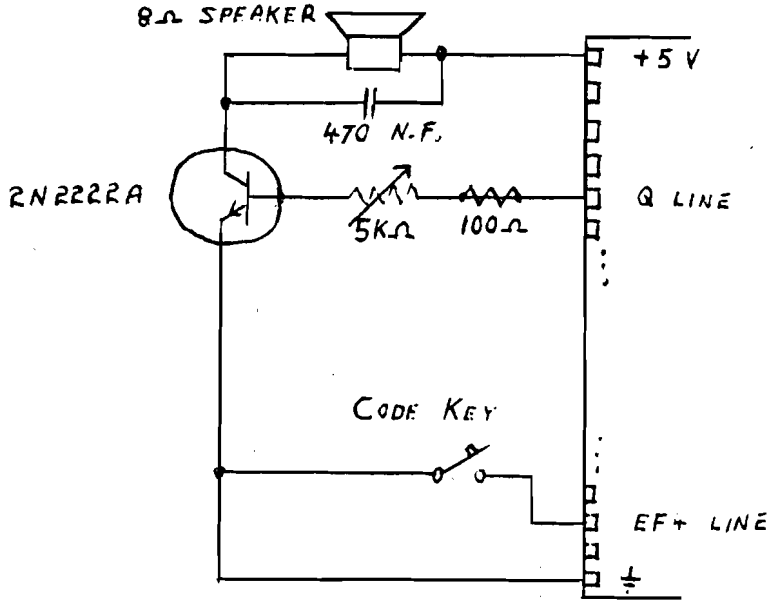
Doug, Nancy Inkster and Doug Olenick

With this short program and a minimum amount of hardware additions, you can turn your micro into the most expensive code practise oscillator you ever saw.

The oscillator can be "keyed" by the "i" button but for true code practise, hook a code key from gnd. to the EF4 line on the edge card.

The freq. can be changed to any one of 256 values, (09 is a good start), by changing the # in memory location 01.

```
00   F8
01   Freq.(09)
02   A7
03   27
04   87
05   3A
06   03
07   39
08   0C
09   7A
0A   30
0B   00
0C   3F
0D   0C
0E   7B
0F   30
10   00
```



## HEX↔ DECIMAL CONVERSION, AND ASCII

Ken Smith

Table 1 is very convenient for hex-decimal conversion. A calculator is not needed. A full address or register is 16 bits or 4 hex digits. The largest hex number is FFFF or 65535 decimal. The 4 HEX=DEC columns in the table corresponds to 4 hex digits in the hex number.

Let a hex number be represented by $X_4X_3X_2X_1$. To convert to decimal, multiply each digit by its weighting factor (4096, 256, 16 and 1 respectively) and add the products. Table 1 has already multiplied each digit by the weighting factors. eg. convert C8F3 to decimal - look up each individual hex digit in the table (in each column) and add the 4 decimal equivalents thus,

C8F3 hex = 49152 + 2048 + 240 + 3
= 51443 decimal

To convert decimal to hex (more often done but more difficult) is the reverse procedure. The decimal number N is broken up such that $N = 4096\ X_4 + 256\ X_3 + 16\ X_2 + X_1$ . Starting with the first hex digit (left), find the decimal number (and corresponding hex) less than or equal to the decimal number N. Subtract the two, giving a remainder. Do the same, using the previous remainder, for all 4 hex digits. eg. convert 40,500 to hex - The largest number less than or equal to 40,500 in the first column is 36864 (or 9000 hex), subtract to get 3636 and so on. Thus,

40,500 = 36864 + 3584 + 48 + 4
= 9E34 hex

Table 2 shows the ASCII code, a universal code used in the computer industry. A full ASCII code has 8 bits (a byte), 7 data bits and 1 parity (usually even). The parity bit is the most significant (left) bit. Table 2 shows ASCII without parity. Hence the maximum code is 0111 1111 or 7F hex. Full ASCII has upper and lower case letters and 32 "control characters" which are usually non-printing. The most important control character is carriage return (0D hex). Many terminals dont have lower case letters; a partial ASCII code. Normal upper-case letters, special characters and a few control characters; at most 64 characters. ASCII code for this is only 6 bits. In fact the PDP-8 text editor "packs" 2 characters into its 12 bit memory locations. The actual form of many special characters varies from terminal to terminal. (I've seen some mighty strange character generator ROMS-you can get ones with various foreign alphabets.)

### TRIVIA

The 1802 has one unusual or illegal opcode; "68". The manual does not say what it does, but here is what it does: "68" is actually an input instruction to port 0 (INP0). The only problem is that during input N2,N1,N0 = 0 (since it is port 0) thus the input port logic is not activated. If none of N2,N1,N0 are high, we can't tell if an I/O instruction is being executed unless "68" is decoded from the data bus during a fetch cycle (50). A "68" inputs data to M(R(X)) (as usual), but puts 68 into D, instead of data into D!

TABLE 1 Hexadecimal - Decimal Conversion

| HEX=BINARY | |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| 8 | 1000 |
| 9 | 1001 |
| A | 1010 |
| B | 1011 |
| C | 1100 |
| D | 1101 |
| E | 1110 |
| F | 1111 |

| HEX=DEC | | HEX=DEC | | HEX=DEC | | HEX=DEC | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 4096 | 1 | 256 | 1 | 16 | 1 | 1 |
| 2 | 8192 | 2 | 512 | 2 | 32 | 2 | 2 |
| 3 | 12288 | 3 | 768 | 3 | 48 | 3 | 3 |
| 4 | 16384 | 4 | 1024 | 4 | 64 | 4 | 4 |
| 5 | 20480 | 5 | 1280 | 5 | 80 | 5 | 5 |
| 6 | 24576 | 6 | 1536 | 6 | 96 | 6 | 6 |
| 7 | 28672 | 7 | 1792 | 7 | 112 | 7 | 7 |
| 8 | 32768 | 8 | 2048 | 8 | 128 | 8 | 8 |
| 9 | 36864 | 9 | 2304 | 9 | 144 | 9 | 9 |
| A | 40960 | A | 2560 | A | 160 | A | 10 |
| B | 45056 | B | 2816 | B | 176 | B | 11 |
| C | 49152 | C | 3072 | C | 192 | C | 12 |
| D | 53248 | D | 3328 | D | 208 | D | 13 |
| E | 57344 | E | 3584 | E | 224 | E | 14 |
| F | 61440 | F | 3840 | F | 240 | F | 15 |

TABLE 2 ASCII (Without Parity)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NUL | SOH | STX | ETX | EOT | ENQ | ACK | BEL | BS | HT | LF | VT | FF | CR | SO | SI |
| 1 | DLE | DC1 | DC2 | DC3 | DC4 | NAK | SYN | ETB | CAN | EM | SUB | ESC | FS | GS | RS | US |
| 2 | SP | ! | " | # | $ | % | & | ' | ( | ) | * | + | , | - | . | / |
| 3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 4 | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 5 | P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ↑ | |
| 6 | ` | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| 7 | p | q | r | s | t | u | v | w | x | y | z | { | ¦ | } | ~ | DEL |

## Control Characters

NUL Null (leader on paper tape)
SOH Start of heading
STX Start of text
ETX End of text
EOT End of transmission
ENQ Enquiry
ACK Acknowledgement
BEL Bell or attention signal
BS Backspace
HT Horizontal tabulation
LF Line feed
VT Vertical tabulation
FF Form feed
CR Carriage return
SO Shift out
SI Shift in
DLE Data link escape
DC1 Device control 1
DC2 Device control 2
DC3 Device control 3
DC4 Device control 4
NAK Negative ack
SYN Synchronous/Idle
ETB End of transmitted block
CAN Cancel (error in data)
EM End of medium
SUB Start of special sequence
ESC Escape
FS File separator
GS Group separator
RS Record separator
US Unit separator
SP Space
DEL Delete

## ITEMS FOR SALE

1. An assembled ELF-II (Netronics R&D Ltd) with power transformer, RCA 1802 User's Manual, RF Modulator and numerous programs from POPULAR ELECTRONICS and IPSO FACTO. Selling price is US $110.00. Please contact: Peter W. Snyder, 1417 E 53rd St., Chicago, Illinois, USA 60615. Phone (312) 241-7236 after 6 PM EST.

2. TEC-1802 (slow chip), with TEKTRON 5V 1A DC regulated power supply and TEC-1802 5 slot mother board. All assembled and with documentation. Contact Mr. M. Skodny, 80 Weir St. S., Hamilton, Ontario, CANADA L8K 3A6.

3. For Sale:

| | | |
|---|---|---|
| 16 pin W.W. gold sockets | - | $ .65 |
| 14 pin W.W. gold sockets | - | .60 |
| 16 pin tin plated solder tail sockets | - | .30 |
| 14 pin tin plated solder tail sockets | - | .25 |
| 50 W.W. Vector pins | - | 2.25 |

Contact: Bernie Murphy, 102 McCraney St., Oakville, Ontario, CANADA L6H 1H6

4. For Trade:
I have 1 Memorex 1240 ASCII terminal; 60, 30, 15, 10 c.p.s., 120 print positions, upper/lower case, paper carrier. A1 condition. Cost $5,500 new. All service manuals included.
I will trade for a 2 channel oscilloscope, 15 MHz or better. I prefer a solid state 'scope, although a good quality TEKTRONIX or HP tube 'scope will be considered. Serious offers only please. Contact: Bernie Murphy, 102 McCrany St., Oakville, Ontario, CANADA L6H 1H6.

## A DIS-ASSEMBLY OF ED MCCORMICK'S MONITOR

Robert Edwards

(Ed. Note: Ed McCormick's Monitor consists of a hex keyboard monitor and cassette interface, in 255 bytes of code which can be stored in a PROM(1702) and located in any memory page. It is designed to work with a 2 MHz clock, and 1K bytes of RAM located at 0400 to 07FF. This monitor was originally published in POPULAR ELECTRONICS and DR. DOBB'S JOURNAL, with accompanying test, in the following form:

| Loc | Contents | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 00 | F8 | 07 | B1 | F8 | FE | A1 | E1 | 6C | 64 | FC | FD | 33 | 3B | 3F | 0D | 37 |
| 10 | 0F | 6C | 64 | 21 | BF | 3F | 15 | 37 | 17 | 6C | 64 | 21 | 21 | AF | F0 | EF |
| 20 | 3F | 20 | 37 | 22 | 32 | 34 | F6 | 33 | 2B | 7B | 6C | 64 | 3F | 2C | 37 | 2E |
| 30 | 31 | 2A | 30 | 2B | DF | 00 | 00 | 00 | 00 | 00 | 00 | 3A | 3E | 7B | F8 | 04 |
| 40 | B1 | F8 | 00 | A1 | B2 | B3 | B4 | B5 | B6 | B7 | 39 | 9E | F8 | 80 | A7 | F8 |
| 50 | 08 | A5 | 7B | F8 | 00 | 51 | 35 | 56 | 3D | 58 | F8 | 00 | FC | 01 | 35 | 5C |
| 60 | FC | ED | 3B | 56 | F8 | B3 | A2 | 7A | 82 | FF | 01 | C4 | 3A | 69 | 35 | 6E |
| 70 | 3D | 70 | F8 | 00 | FC | 01 | 35 | 74 | FC | ED | 3B | 85 | 7A | F0 | F6 | 51 |
| 80 | F8 | 59 | A2 | 30 | 8F | 7B | F0 | F6 | 51 | 87 | F4 | 51 | F8 | 68 | A2 | 85 |
| 90 | FF | 01 | A5 | 3A | 68 | 64 | 91 | FF | 08 | 3A | 4F | 7A | 00 | 00 | 7A | F8 |
| A0 | 00 | A6 | A7 | F8 | 10 | A2 | F8 | 01 | A3 | 30 | AB | 31 | B0 | 7B | 30 | B3 |
| B0 | 7A | 30 | B3 | 83 | FF | 01 | 3A | B4 | 82 | FF | 01 | A2 | 32 | C8 | F8 | 07 |
| C0 | FF | 01 | 3A | C0 | 30 | C6 | 30 | AB | 86 | 3A | D7 | 3F | A3 | F8 | 01 | A6 |
| D0 | F0 | A4 | F8 | 09 | A5 | 30 | E5 | 87 | 3A | F4 | 85 | FF | 01 | A5 | 32 | ED |
| E0 | 84 | 76 | A4 | 33 | A3 | F8 | 08 | A2 | F8 | 0E | A3 | 30 | AB | F8 | 01 | A7 |
| F0 | F8 | 20 | 30 | A5 | F8 | 00 | A7 | 64 | 91 | FF | 08 | 32 | 9E | 30 | D0 | 00 |

I leave it up to you, the reader, to decide which form of software documentation you prefer: the hex dump above, or the commented assembly language below.)

EDWARD MCCORMICS'S ELF OPERATING SYSTEM (2 MHZ CLOCK) & CASSETTE INTERFACE FOR 1K MEMORY (0400 TO 07FF)

Robert Edwards

```
00   F8 07   START    LDI   07        07FE → R1; 07FE = ADDRESS OF FUNCTION
02   B1               PHI   1                  07FF = ADDRESS OF HIGH & LOW BYTES OF ADDRESS PUT IN
03   F8 FE            LDI   FE                                                    THROUGH SWITCHES
05   A1               PLO   1
06   E1               SEX   1         1 → X
07   6C               INP   4         GET SWITCHES (FUNCTION REQUESTED)
08   64               OUT   4         PUT SWITCHES ON DISPLAY
09   FC FD            ADI   FD        D OVERFLOWS IF SWITCHES > 2
0B   33 3B            BD    CASIO     IF SWITCHES > 2 GO TO CASSETTE I0
0D   3F 0D            BN4   *         WAIT FOR TOGGLE
0F   37 0F            B4    *         WAIT UNTIL TOGGLE GOES OFF
11   6C               INP   4         GET SWITCHES (HIGH BYTE OF ADDRESS)
12   64               OUT   4         PUT HIGH BYTE ON DISPLAY
13   21               DEC   1         MOVE R1(X) BACK TO 07FF
14   BF               PHI   F         PUT HIGH BYTE OF ADDRESS IN RF.1
15   3F 15            BN4   *         WAIT FOR TOGGLE
17   37 17            B4    *         WAIT UNTIL TOGGLE GOES OFF
19   6C               INP   4         GET SWITCHES (LOW BYTE OF ADDRESS)
1A   64               OUT   4         PUT LOW BYTE ON DISPLAY
1B   21               DEC   1         MOVE R1(X) BACK TO 07FF
1C   21               DEC   1         MOVE R1(X) BACK TO 07FE
1D   AF               PLO   F         PUT LOW BYTE OF ADDRESS IN RF.0
1E   F0               LDX             REQUESTED FUNCTION → D
1F   EF               SEX   F         F → X
20   3F 20            BN4   *         WAIT FOR TOGGLE
22   37 22            B4    *         WAIT UNTIL TOGGLE GOES OFF
24   32 34            BZ    EXEC      SWITCHES=0 → START EXECUTION AT USERS ADDRESS
26   F6               SHR             LOW BIT OF D INTO DF
27   33 2B            BD    DISLOOP   SWITCHES=1 → DISPLAY MEMORY AT USERS ADDRESS
29   7B               SEQ             LIGHT LED TO SIGNAL ALTERING MEMORY
2A   6C       ALTLOOP INP   4         GET SWITCHES
2B   64       DISLOOP OUT   4         NEW DATA FOR THIS MEMORY LOC TO DISPLAY [R1(X) = R1(X)+1]
2C   3F 2C            BN4   *         WAIT FOR TOGGLE
2E   37 2E            B4    *         WAIT UNTIL TOGGLE GOES OFF
30   31 2A            BQ    ALTLOOP   ALTER MEMORY AT SUCCEEDING R1(X)
32   30 2B            BR    DISLOOP   DISPLAY MEMORY AT NEXT SUCCEEDING R1(X)
34   DF       EXEC    SEP   F         START USERS PROGRAM WITH P = F
```

```
3B   3A 3E  CASIO     BNZ   *+3
3D   7B               SEQ                 SET Q TO BRANCH TO CASSETTE WRITE AFTER INITIALIZATION
3E   F8 04            LDI   04            0400 → R1
40   B1               PHI   1
41   F8 00            LDI   00
43   A1               PLO   1
44   B2               PHI   2             CLEAR HIGH BYTE OF R2 THROUGH R7
45   B3               PHI   3
46   B4               PHI   4
47   B5               PHI   5
48   B6               PHI   6
49   B7               PHI   7
4A   39 9E            BNQ   WRITECAS
4C   F8 80  READCAS   LDI   80            PUT HIGH ORDER BIT IN R7 (SET END OF BYTE SWITCH TRUE)
4E   A7               PLO   7
4F   F8 08  RCSTART   LDI   08            PUT SHIFT COUNT IN RS
51   A5               PLO   5
52   7B               SEQ                 TURN OF Q
53   F8 00            LDI   00            CLEAR BYTE TO BE LOADED
55   51               STN   1
56   35 56  IBMLOOP   B2    *             LOOP ON MARKS IN TAPE RECORDER OF BETWEEN BYTES
58   3D 58            BN2   *             START 208.33333μs HALF CYCLE MEASURE (EQUALS 52 4μs CYCLE TIMES)
5A   F8 00            LDI   00
5C   FC 01            ADI   01            ADD ONE TO D EVERY 16μs (EVERY 4 CYCLE TIMES)
5E   35 5C            B2    *-2           MARK EXITS WITH D EQUALS 12 AT MOST
60   FC ED            ADI   ED            WAS EF2 HIGH FOR 224μs (56 4μs CYCLE TIMES)(13+ED OVERFLOWS D)
62   3B 56            BND   IBMLOOP
64   F8 B3            LDI   B3            IF SPACE START 5 MILLISECOND DELAY
66   A2               PLO   2
67   7A               REQ
68   82       BITLOOP GLO   2             DELAY ACCORDING TO DELAY CONSTANT IN R2.0
69   FF 01            SMI   01              R2.0 = B3 (179) FOR 5 MILLISECONDS; 68(104) FOR 2.9 MS
6B   C4               NOP                          59 (89) FOR 2.5 MILLISECONDS
6C   3A 69            BNZ   *-3             LOOP IS 7  4μs CYCLE TIMES
                                              179 X 28 = 5012 μs; 104 X 28 = 2912μs
                                                       89 X 28 = 2492μs
```

```
35 6E            B2    *          DETERMINE IF A MARK OR
3D 70            BN2   *            SPACE AT SAMPLING TIME
F8 00            LDI   00         WAIT FOR EF2 TO GO HIGH THEN
FC 01            ADI   01         ADD 1 TO D FOR EVERY 4 CYCLE TIMES (16 s)
35 74            B2    *-2          LOOP BACK IF EF2 STILL HIGH
FC ED            ADI   ED         D OVERFLOWS IF D>12  (12 = 4 + 4 X 12 = 52 C.T. ≃ 208μs)
3B 85            BND   GOTMARK
7A       GOTSPACE REQ              GOT A SPACE - TURN Q OFF
F0               LDX                GET PARTIAL BYTE
F6               SR                 SHIFT IN A ZERO FROM TOP
51               STN   1            PUT BACK PARTIAL BYTE
F8 59            LDI   59H        SET DELAY FOR 2.5 MILLISECONDS
A2               PLO   2
30 8F            BR    NEXTBIT
7B       GOTMARK  SEQ              GOT A MARK - TURN Q ON
F0               LDX                GET PARTIAL BYTE
F6               SR                 SHIFT IT RIGHT ONE BIT
51               STN   1              STORE IT
87               GLO   7          GET CONSTANT 80 OUT OF R7.0
F4               ADX                ADD IT TO PARTIAL BYTE
51               STN   1              STORE IT
F8 68            LDI   68         SET DELAY FOR 2.9 MILLISECONDS
A2               PLO   2
85       NEXTBIT  GLO   5          GO BACK TO BITLOOP UNTIL 8 BITS HAVE
FF 01            SMI   01             BEEN PROCESSED
A5               PLO   5
3A 68            BNZ   BITLOOP
64               OUT   4          DISPLAY BYTE
91               GHI   1
FF 08            SMI   08         07FF IS LAST BYTE TO BE LOADED
3A 4F            BNZ   RCSTART
7A               REQ              IF ALL OF CORE LOADED - TURN OF Q AND IDLE
00               IDLE
00
7A       WRITECAS REQ              TURN Q OFF
F8 00            LDI   00
A6               PLO   6          SET R6 - LEADER SWITCH
A7               PLO   7          CLEAR R7 - END OF BLOCK SWITCH
F8 10    MARKGEN  LDI   10         GENERATE A MARK - 8 CYCLES AT 2400 HZ
A2               PLO   2            208.3333μs HALF CYCLE (52 4μs CYCLE TIMES)
```

```
A6   F8 01          LDI  01
A8   A3             PLO  3
A9   30 AB          BR   *+2
AB   31 B0  QTEST   BQ   RESETQ     54 WORD TIMES TO HERE FROM EITHER SEQ or REQ FOR MARK
AD   7B             SEQ                106 TO HERE FOR SPACE
AE   30 B3          BR   VARDELAY   54 X 4 = 216 (SHOULD BE 208.3333); 106X4 = 424(SHOULD BE 416.6667)
B0   7A     RESETQ  REQ
B1   30 B3          BR   *+1
B3   83     VARDELAY GLO 3          6 WORD TIMES IN ½ CYCLE TO HERE
B4   FF 01          SMI  01
B6   3A B4          BNZ  *-2        10 TO HERE FOR MARK - 62 FOR SPACE
B8   82             GLO  2          DECREMENT CYCLE COUNTER
B9   FF 01          SMI  01
BB   A2             PLO  2
BC   32 C8          BZ   LEAD TEST  18 TO HERE FOR MARK - 70 FOR SPACE
BE   F8 07  FIXDELAY LDI 07         FIXED..DELAY OF 28 CYCLE TIMES IN LOOP (SHOULD BE LDI 05)
C0   FF 01          SMI  01                                                (26 CYCLE TIMES)
C2   3A C0          BNZ  *-2
C4   30 C6          BR   *+2        2 CYCLE DELAY (SHOULD BE 2 NOPS)
C6   30 AB          BR   QTEST      52 TO HERE FOR MARK - 104 FOR SPACE
C8   86     LEADTEST GLO 6          LEADER SWITCH IS ZERO WHEN
C9   3A D7          BNZ  EBTEST       LEADER IS BEING GENERATED
CB   3F A3          BN4  MARKGEN    IF NO TOGGLE - CONTINUE MARK GENERATION
CD   F8 01          LDI  01         CLEAR R6 - LEADER FLAG
CF   A6             PLO  6
D0   F0     BYTEOUT LDX             PUT BYTE TO BE OUTPUT IN R4.0
D1   A4             PLO  4
D2   F8 09          LDI  09         PUT SHIFT COUNT IN R5.0
D4   A5             PLO  5
D5   30 E5          BR   SPACEGEN   GENERATE SPACE THAT PRECEEDS DATA
D7   87     EBTEST  GLO  7          BRANCH AFTER DOUBLE MARKS AT END OF BYTE
D8   3A F4          BNZ  ENDBYTE
DA   85             GLO  5          REDUCE SHIFT COUNT - BRANCH IF ALL
DB   FF 01          SMI  01           8 BITS OUTPUT
DD   A5             PLO  5
DE   32 ED          BZ   DOUBMARK
E0   84             GLO  4          GO TO MARK OR SPACE ACCORDING TO
E1   76             SHRC              BIT OF BYTE
E2   A4             PLO  4
E3   33 A3          BD   MARKGEN
E5   F8 08  SPACEGEN LDI 08         GENERATE A SPACE - 4 CYCLES AT 1200 HZ
```

```
A2                  PLO  2
F8 0E               LDI  0E          VARIABLE DELAY FOR SPACE - 56 CYCLE TIMES VS 4 FOR MARK
A3                  PLO  3                 (THIS SHOULD BE LDI 0D??)
30 AB               BR   QTEST
F8 01  DOUBMARK     LDI  01          AT END OF BYTE - SET EOB SWITCH
A7                  PLO  7              AND GENERATE DOUBLE MARK
F8 20               LDI  20
30 A5               BR   MARKGEN+2
F8 00  ENDBYTE      LDI  00          AFTER DOUBLE MARK AT END OF BYTE -
A7                  PLO  7             RESET END OF BYTE SWITCH - DISPLAY
64                  OUT  4                BYTE
91                  GHI  1
FF 08               SMI  08          ALL OF CORE WRITTEN AFTER 07FF
32 9E               BZ   WRITECAS      GENERATE TRAILER OF MARKS
30 D0               BR   BYTEOUT     DO ANOTHER BYTE
```

## NOTE OF CAUTION ON USING MY PROGRAMS FOR CASSETTE INTERFACE WITH CLOCK RATES LESS THAN 2 MHZ

Edw. M. McCormick



The KC standard interface described in Feb. '78 Popular Electronics and No. 19 of Dr. Dobbs urnal was designed for a 2 mhz crystal. Although the programs will operate at lower clock fre-encies, they will not produce the KC standard tones. If a 1.78 mhz crystal is used the tones ll be 1030 hz and 2060 hz. A 1 mhz clock means that these tones are 597 hz and 1158 hz. How-er, the cassette read and write programs should operate correctly at these lower frequencies. will simply take longer to read and write the records. The only problem will be in exchanging pes between systems.

Reexamination of these programs (written a year ago) indicates that it is not possible to dify them to get 1200 hz and 2400 hz tones if the clock is less than 2 mhz. The basic reason r this is in the cassette write program. The minimum number of steps that are used for each lf cycle (in order to make all the necessary Checks) is 27 instruction executions. This takes 6 microseconds with a 2 mhz crystal. Thus a half cycle on Q can not be less than this time ich means that 2315 hz is the highest tone which can be generated with that crystal. To obtain gher frequencies for lower clock rates would require a complete rewrite of the programs. It n not be done by changing any of the values in the programs (including the operating system on ge 34 of DDJ).

Either use a 2 mhz crystal or live with the non-standard tones when using lower frequency ystals.

Sorry about that!

# THE 1802 MUSIC MACHINE

by Cec Williams

It is well known by most experimenters, that music is made up of different frequencies and durations, plus pauses (or rests),of different durations. It is also known that a microprocessor is eminently capable of deriving such frequencies, and outputting them from a serial output port. In the case of the CDI802 microprocessor, by R.C.A., as used in the Cosmac Elf, the Q-output port can be programmed to play a melody, via any suitable amplifier and loudspeaker, or even 10 - 20" of wire!

Several music programs were studied, including those in the Nov.1977 Interface Age, and in the Feb.1978 Popular Electronics. While they worked well, within their limitations, it was decided to make a program with some features that would be more flexible. This would enable even a neophyte musician the knowledge to program in his own melodies. Some useful features of the program are:

1. Only one memory byte is needed to generate the note frequency, its duration, and whether it is in a low or high octave.
2. By sensing the end of a melody, it can be repeated over.
3. A range of two octaves, with sharps and flats (the black notes of a piano), may be programmed.
4. Once the program has a melody entered at the basic speed, a simple change can alter the speed to any one of four other speeds.
5. As music itself is based on a code system of seven letters, A to G, (the white notes on a piano), six of these seven notes, A to F, are coded with the same HEX-code letters. Note G, and the five accidentals,(sharps or flats), use HEX-code numbers. A pause, or rest, is always coded as a zero.



## THE MUSIC BYTE

Each data byte that is used to program a melody, has its 8 data bits broken up into 3 parts, as shown in Figure 1. The 4 low bits define the note frequency, the next 3 bits define the duration, while the high bit defines which octave is chosen. The resultant data byte is called a Music Byte.

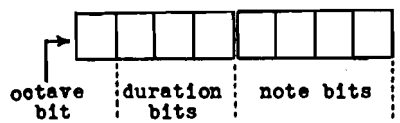



Figure 1

**Note Bits:**
Whether the note to be coded is in the high or the low octave, the 4 note bits are always coded in HEX-code as the lower HEX-digit, shown in Table 1.

| | | | | | | | | $B^{\flat}$ | $D^{\flat}$ | $E^{\flat}$ | $G^{\flat}$ | $A^{\flat}$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Music Code: | A | B | C | D | E | F | G | $A^{\#}$ | $C^{\#}$ | $D^{\#}$ | $F^{\#}$ | $G^{\#}$ | rest |
| Low Hex digit : | A | B | C | D | E | F | 9 | 4 | 5 | 6 | 7 | 8 | 0 |

Where a # is a sharp, a half-tone above the basic note, and a $\flat$ is a flat, a half-tone below the basic note.

Table 1

**Duration Bits:**
For the duration of a note, music works in a binary fashion. Some basic common notes are quarter notes, eighth notes, sixteenth notes,etc., known as crotchets, quavers and semiquavers, respectively. They sometimes have a dot placed after them, which increases their duration by 50%. Such an increase can easily be coded in a binary system. Some examples are shown in Table 2.

| Music Code | Duration | Duration Bits |
|---|---|---|
|  | 1/16-note | 0 0 1 |
|  | 1/8-note | 0 1 0 |
|  | 3/16-note | 0 1 1 |
|  | 1/4-note | 1 0 0 |
|  | 3/8-note | 1 1 0 |
|  | 1/8-rest | 0 1 0 |
|  | 1/4-rest | 1 0 0 |

Table 2

**Octave Bit:**
The program has one basic low octave of twelve semitones programmed into it. This octave includes the seven basic notes labelled A to G, plus the five sharps or flats. To derive the high octave of notes, the program doubles the pitch of these low notes. This is triggered by programming a '1' in the octave bit. Some examples show how the music byte is built up, and then converted to HEX-code:

1. Note G, 3/8-note long,high octave: 1 1 1 0 1 0 0 1 = E9
2. Note B, 1/8-note long, low octave: 0 0 1 0 1 0 1 1 = 2B
3. A rest, of duration 1/4-note: 0 1 0 0 0 0 0 0 = 40

## THEORY OF OPERATION

The program shown in Table 4 is walked-through in some detail below. This has been done, to help neophyte programmers like myself and others, to be better able to grasp the use of some of the simple, and extremely useful 1802-program code instructions. To aid in these explanations, an arbitrary first Music Byte of B9, (a high octave G, 3/16-note long) has been placed in address 50, the address of the first note of a melody. The 1802-register locations are variously referenced as R-1.0, low-C, high-B,Register-F, etc.

### Initial Parameters

Register-8 is firstly made the stack pointer. Next,address 50 is loaded into low-8. The speed modifier address, 43, is then loaded into low-9. The LDXA instruction then extracts the contents of address 50, (which in the example is Music Byte B9), and puts it in the D-accumulator. The PLO A instruction then stores this B9 in low-A. With a Music Byte of 00 put at the end of a melody, the program branches back to address 01, and repeats the whole melody. This will continue, until a 'Halt' command stops the program.

### Duration Constant

This is obtained by ANDing 70 with the Music Byte. This masks

out both the octave bit, and the note bits, leaving the duration byte, 30. As this is stored in high-B, the complete constant is 30 00. This is equivalent to a decimal number of:

3 x 16 x 16 x 16 = 12,288

This constant is later used in the duration loop,where it will be decremented, while the Q-output is oscillating at the note frequency,until hi-B gets to zero. This gives the note's duration.

Speed Modifier

As written, with NOP's from address OE to 16, the speed of the melody is based on the timing the 'composer' puts into each Music Byte, and remains unaltered. Later, it will be shown how to modify the overall speed of the melody, either up or down, by editing some or all of these nine address locations.

Note Constant

The note bits are firstly extracted from the Music Byte by ANDing it with OF. This masks out the duration bits and the octave bit, leaving a note constant of 09. This constant is then OR'd with 40, which produces the address of the note's frequency constant, associated with the particular note itself. Here, 09 OR'd with 40 gives 49, the address in which the frequency constant for note G had been previously inserted. The LDN instruction gets this frequency constant, 33 (or decimal 51); PLO E stores it in low-E.

(Note that by ANDing with 40, all reference notes used in the program are located within addresses 40 through 4F).

Octave Constant

The Music Byte is again retrieved by GLO A, and AND'd with 80. This masks out the note and duration bits, and in our example, leaves a '1' in the octave bit. This gives an octave constant of 10. (If a low octave note had been programmed instead, an 00 would have resulted, which would cause the high octave generator to be bypassed.) However, it is needed in our example, so the frequency constant is retrieved, and a Shift Right, SHR, is made. This divides 33 (or decimal 51) by two. This produces 19 (or decimal 25), also DF = 1 ,( a remainder). Then a '1' is subtracted from the D-register,which is then saved as 18 in low-E. This gives a truer pitch for the high octave notes.

Timing Loops:

1. The Note, or Q On/Off Loop

A copy is made of the duration constant in high-B, and put in high-F. Next, the duration constant from high-F is put in the loop, and checked to see if it has been decremented to zero, at which point, a branch would be made to pick up the next note. As high-F is not yet zero, the frequency constant is put in the D-accumulator, where it is decremented by subtracting 01. Then, the duration constant in register-F is decremented, and a branch is made back to address 31. This short inner loop is travelled around, until the frequency constant in the D-accumulator has reached zero.The note constant in low-C is checked next to see if it contains either a rest (00) or a note constant. If it had been a rest, then the BZ check at address 37 would bypass the succeeding Q-state changes. Thus, Q would remain stationary (either On or Off), enabling a programmed rest, or pause, to be timed out. In the example, with low-C not equal to zero, a BQ check is made to see if Q = 1. If it is, a branch is made to reset Q = 0, and then back to address 2D. But, if Q = 0 at address 39, the next instruction sets Q = 1, followed by a branch to 2D. This process repeats itself, to continuously switch Q on and off.



11. The Duration Loop

While the above is going on, each time the Q-output is switched on and off, the counter at register-F is decremented many times; (it decrements 18 (or decimal 24) times, for each phase of Q-on, or Q-off). After sufficient of these Hex 18 decrements have decreased the contents of register-F, eventually high-F becomes zero. Then, the next time the program reaches address 2E, the BZ check finds the duration timer has run out, so a branch is made to pick up the next note.

It can now be seen that Q switches on and off, at a rate set by the initial low-E, and a duration set by the initial high-F.

Duration, or Speed Modifier

The overall speed of the melody may be too fast or too slow for the melody programmed in. However, a simple change of speed can be made, by modifying the program at address OE to either SHR or SHL. These two instructions take the contents of the D-accumulator, (in our example, Hex 30), and shift the binary digits one place to the right, or the left, as desired. Thus:

30 = 0011 0000 (count=48) : 30 = 0011 0000 (count=48)
SHR = 0001 1000 = 18 (count=24) : SHL = 0110 0000 = 60 (count=96)

If either instruction is used, followed by a save instruction PHI B at address at OF, the duration constant, and hence the melody speed, will be either doubled or halved, as applicable.

One of the other two alternate speeds can also be programmed into the same addresses. One of these multiplies the original duration constant in register-B by 1.5, and the other by 0.75. Addresses 04 to 06, plus 43, are programmed to temporarily designate register-9 as a stack pointer,only during this speed modifier. The four modifiers are shown in Teble 3. An explanation follows, of the 0.75 duration modifier, with Hex 30 being the original contents of high-B:

| Memory Location | Half Duration | Double Duration | 3/4 x Duration | 3/2 x Duration |
|---|---|---|---|---|
| OE | F6 (SHR) | FE (SHL) | FE (SHL) | FE |
| OF | BB (PHI B) | BB | 59 (STR 9) | 59 |
| 10 | C4 | C4 | E9 (SEX 9) | E9 |
| 11 | C4 | C4 | 9B (GHI B) | 9B |
| 12 | C4 | C4 | F6 (SHR) | F6 |
| 13 | C4 | C4 | F5 (SD) | F5 |
| 14 | C4 | C4 | F6 (SHR) | C4 (NOP) |
| 15 | C4 | C4 | BB (PHI B) | BB |
| 16 | C4 | C4 | E8 (SEX 8) | E8 |

Table 3

FE doubles the 30 to 60
59 stores this 60 at address 43
E9 sets X to 9, making register-9 a temporary stack pointer
9B retrieves the original counter of 30
F6 halves the 30, giving Hex-18
F5 subtracts this 18 from 60 in address 43, giving Hex-48
F6 halves the 48, giving 24
BB stores this modified duration constant back in high-B
E8 resets the stack pointer to 8, for the next Music Byte

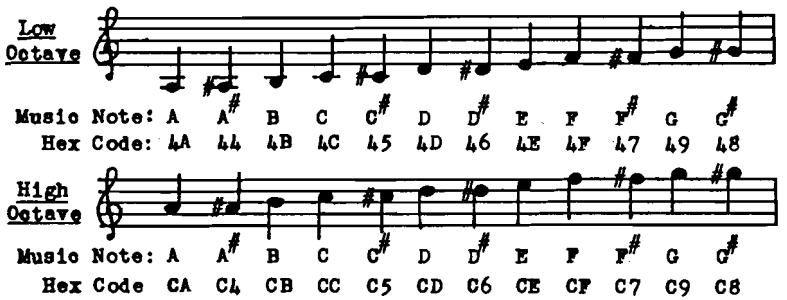
Thus, the original duration counter of 30 (or decimal 48), has been changed to 24 (or decimal 36), which is 0.75 times its original value. This means that every duration of note or rest will be lowered, giving an overall speed increase of 4/3 times.

## MELODY PROGRAMMING

Now comes the interesting task of programming melodies of one's own choice, and then being able to sit back and listen to them, or accompany them on a guitar, or piano, etc.

In Figure 2, the two lines of notes show both octaves that are programmable by the Music Machine. The two lines are prefixed by a symbol called the G-clef. This represents the upper clef of written music, in which most melodies, often with accompanying words, are written.

The pitch, or frequency of the notes, is coded by the same Hex-digit, for either the high or low octave note. If a rest (or pause) needs coding, this lower Hex-digit is always coded as a zero. To differentiate between these two octaves, a '1' in bit-8 of the Music Byte changes the frequency to a high octave note.




Left-Hand Hex-Digit Options: Figure 2

| | 1/16 Note | 1/8 Note | 3/16 Note | 1/4 Note | 5/16 Note(rare) | 3/8 Note | 7/16 Note(rare) |
|---|---|---|---|---|---|---|---|
| Low Octave: | 1 | 2 | 3 | 4 | (5) | 6 | (7) |
| High Octave: | 9 | A | B | C | (D) | E | (F) |

Note: A half note, 𝅗𝅥 called a minim, is split between two music bytes.
A dotted minim, 𝅗𝅥. , needs three music bytes.
A whole note , 𝅝 , called a breve, needs four music bytes.

The two octaves of notes shown in Figure 2, have all been coded with a duration of '4' for a 1/4-note, (a crotchet). Below these notes is a summary of the left-hand Hex-digit options; the only difference between the two rows being, of course, the fact that the high-octave digits are decimal-8 greater than the low-octave digits. This is due to the octave-bit of '1' in the highest bit of the Music Byte.

### Melody No.1

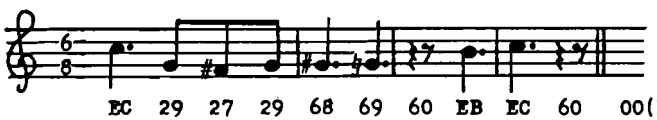



Figure 3

Figure 3 shows a simple ending to a tune. It is written in the key of C, in 6:8 time. This timing means the equivalent of 6 notes to a bar, each being 1/8 of a whole note in duration. The tune could have been written twice as fast, using a B and 3, instead of E and 6, and a 1 instead of 2, in the first Hex-digit. Note that the sixth note has a new accidental after it, called a 'natural'. It cancels the G# in the same bar, and the note reverts back to G-natural, or just G.



For longer melodies, one of the Speed Modifiers would be put in addresses 0E to 16, as a much easier way to adjust the speed of the melody. Note that the two rests, of duration 4/16 and 2/16, are combined as duration 6/16, to save precious memory space.

### Melody No.2

The following melody is part of a well known tango. It has been written directly in Hex-code, and should be recognised by almost everyone:

46 2D 40 26 2D 40 26 2D 27 AA 60 46 2D 40 26 2D 40 26 2D 29 A4 60

46 2D 40 26 2D 40 26 2D 27 AA 60 C4 29 40 A6 AD AC 20 94 9A 29 20

4D 29 20 00

Without going any further into music theory, the few introductions into basic music given herein, should be enough to enable a microprocessor experimenter to transcribe a favorite melody into Music Bytes. A musician friend could give assistance, if difficulties are encountered in early attempts at transcription.

### Hardware Considerations

To hear the melody, the output at the Q-output port must be made audible. A simple way of achieving this, is to connect a short wire to the Q-output, and bring the wire close to the antenna of an A.M. radio. As long as the 1802-chip clock frequency lies in the range of, say, 750kHz to 1600kHz, the radio can be tuned to the clock frequency, and the melody can be heard.

In the author's system, the original 1MHz clock frequency had been increased by installing a 2.01MHz crystal. This simple change enabled reasonably accurate high frequency notes to be generated by the Music Machine. With this crystal, the low-octave note A is about 220Hz, and the high-octave G# is about 784Hz. These frequencies, and all the intermediate ones, have been made as close as possible to those of a properly tuned piano. To accomodate the higher clock frequency, a simple audio 1-transistor amplifier was used, as shown in Figure 4. Any transistor of $\beta \geq 50$ is O.K. The speaker can be 8 ohms or higher. A ratio of 20:1 for T1 works well.

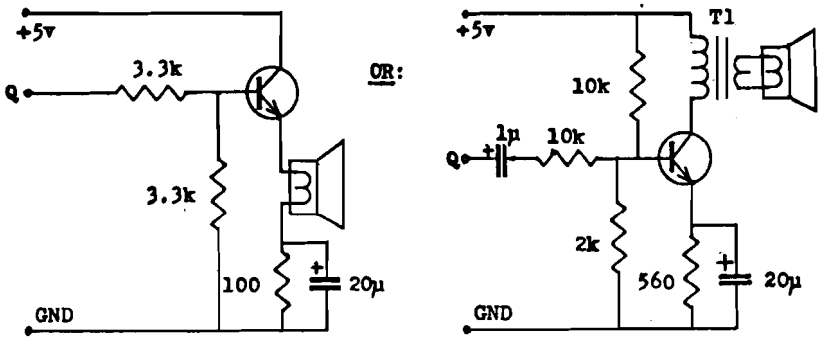



Figure 4

However, the use of a 1MHz clock frequency will not affect the operation of the program. The basic difference will be that the note's frequencies will be halved, and their durations will be doubled. The notes will still be in tune, except that they would be one octave lower than with a 2.01MHz clock frequency. The speed difference can be adjusted, as desired, by using a speed modifier.

## Derivation of the Frequency Constants

### The Timing Loops

Each instruction in the loops shown in Figure 5 takes a total of 16 machine cycles to complete.

To program the Q-output to switch ON and OFF at a specific frequency of 'P' Hz, a frequency constant 'Y' must be derived. This is a decimal constant, whose equivalent in Hex-code is put in low-E. The D-accumulator will be decremented Y-times in the inner loop 'X'.

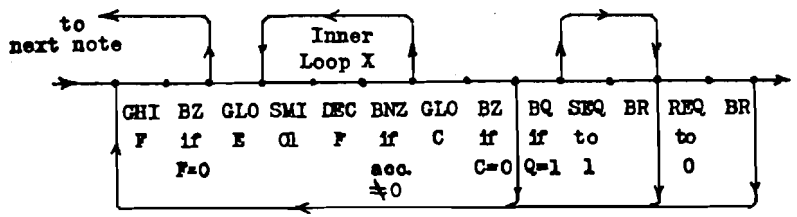



Figure 5

So, low-E = $Y_{decimal}$

Let K = clock frequency in MHz

Then, t = time of 16 machine cycles = $\frac{16}{K}$

Analysis of the overall loop gives:

Time Q is switched ON = t + 3t + 3tY + 4t = 8t + 3tY

Time Q is switched OFF = t + 3t + 3tY + 4t = 8t + 3tY

Time of 1 Q-ON/OFF cycle = T usec = 16t + 6tY

This gives; $T = t(16 + 6Y)$

But, $T = \frac{10^6}{P}$ usec, where P = note frequency in Hz

So, $\frac{10^6}{P} = t(16 + 6Y)$ Or: $6Y = \frac{10^6}{tP} - 16$ Or: $Y = \frac{10^6}{6tP} - \frac{8}{3}$

Substituting $t = \frac{16}{K}$, then: $\boxed{Y = \frac{10^6 K}{96P} - 2.67}$

Example:

Assume K = 2.01 MHz; find Y for low-octave G of 392 Hz

Then, $Y = \frac{2.01 \times 10^6}{96 \times 392} - 2.67 = 50.7$ (or 51 as a whole number)

Converting to Hex, $Y = 33_{Hex}$

The constant 33 is located in address 49 as the frequency constant for low-octave G of 392 Hz. Similar constants can be derived for any clock frequency or any note frequency.

(The actual note frequencies were obtained from:"Reference Data for Radio Engineers", by I.T.& T.Corp.)

### Derivation of the Duration Constants

A duration constant of 30 00 in Register-F will be decremented 3 x 16 x 16 x 16 times, to 00 00. In our case, we only need high-F to reach zero.

To get high-F to 00, number of decrements = 12288 - 255 = 12033

During 1-cycle of low-octave-G, the D-accumulator is decremented to zero twice, giving 2 x 51 or 102 decrements.

Now, every time D-accumulator is decremented, so is Register-F.

So, high-F lets Q cycle at 392Hz, or: $\frac{12033}{102} = 117.97$ (= 118times)

But, one cycle of a G of 392Hz is $\frac{10^6}{392} = 2551$ usec

So, duration 30 is equivalent to 118 x 2551 usec = .301 sec

Similar calculations show that duration 10 = .10 sec
20 = .20 sec
40 = .41 sec
60 = .61 sec

Check:

Melody No.2, shown earlier, has the equivalent of 32 1/4-notes in it, each of Hex-duration 40. When played through five times for a total of 160 1/4-notes, the total time taken was 65 seconds.

This gave a duration for '40' time of $\frac{65}{160}$ or 0.41 seconds. Thus,the theory agreed with practice.

### OTHER USES

Another use for this program, with two minor modifications, is to employ it as an audio sweep generator. For this purpose, the codes at addresses 2A, 2C, 2F, 42 and 50 are used. The two modifications are:

At address 2A, change C4 to 2E (decrement E)
At address 2F, change 04 to 2A (a new branch address)

In the first Music Byte address of 50, a "Sweep Byte" of 22 is inserted as a first attempt. If the program is now run, it will be seen that 22 is put in low-A; low-B becomes 20; low-C is 02; low-D is 42, (which makes low-E a value of FF from address 42). This low-E of FF is decremented to FE at address 2A, ( or decimal 254). After copying to low-F, the frequency and duration loops are picked up, letting the Q-output oscillate at 81.68 Hz for about 0.2 seconds. The program then branches back to address 2A, where low-E is again decremented, dropping to FD, or decimal 253. This allows Q to oscillate at 82.00 Hz for the next 0.2 seconds. The process repeats, with low-E dropping and the frequency increasing, until low-E reaches 01. Then, the next decrement changes it to 00, and the sweep starts up again at the low frequency. With the author's 2.01 MHz clock frequency, the highest frequency obtainable is about 9 kHz.

It can be seen that as the frequency gets higher, the steps between notes become more apparent, as low-E gets smaller and smaller. That is to say, a decrement of 01 when low-E is 250, only varies the frequency by about 0.4%. However, when low-E changes from 9 to 8, a frequency change of about 10% occurs.

With the "Sweep Byte" of 22 in address 50, the total duration for one complete sweep is about 52 seconds. Any "Sweep Byte" from 12 to 72 may be put in address 50; (it must end in a 2). The sweep duration can thus be programmed to last from about 26 secs. to over 3 minutes. The Speed Modifiers of doubling or halving the duration can also be brought into play, if desired.

A possible practical use of this sweep generator, is in the testing of a complete audio system. With an oscilloscope on the output of the amplifier, the response to the square-wave output from the Q-port can be examined. With a loudspeaker on the amplifier output, resonances, particularly at the lower frequencies, can be detected.

In conclusion, it is felt that this reasonably simple program will stimulate experimenters to work out further modifications.

Table 4

| MEMORY LOCN. | OBJECT CODE | MNEMONIC STATEMENT | REGISTER'D' | COMMENTS |
|---|---|---|---|---|
| 00 | E8 | SEX 8 | | Use Register-8 as Stack-Ptr. |
| 01 | F8 50 | LDI 50 | 50 | 1st Music Byte loc'n to R-8.0 |
| 03 | A8 | PLO 8 | 50 | |
| 04 | F8 43 | LDI 43 | 43 | Speed modifier loc'n to R-9.0 |
| 06 | A9 | PLO 9 | 43 | |
| 07 | 72 | LDXA | B9 | Get Music Byte;advance stk.ptr. |
| 08 | AA | PLO A | B9 | Store Music Byte in R-A.0 |
| 09 | 32 01 | BZ | B9 | If Mus.Byte is 00,repeat tune |
| 0B | FA 70 | ANI 70 | 30 | AND 70 with B9 |
| 0D | BB | PHI B | 30 | Store result-duration byte = 30 |
| 0E to 16 | A11 C4 | NOP | | (See text for Speed Modifier) |
| 17 | 8A | GLO A | B9 | Get Music Byte |
| 18 | FA 0F | ANI 0F | 09 | AND 0F with B9 |
| 1A | AC | PLO C | 09 | Store result-note constant = 09 |
| 1B | F9 40 | ORI 40 | 49 | OR 40 with 09 |
| 1D | AD | PLO D | 49 | Store result-note loc'n = 49 |
| 1E | 0D | LDN | 33 | Get freq.constant in loc'n 49 |
| 1F | AE | PLO E | 33 | Store note freq.constant = 33 |
| 20 | 8A | GLO A | B9 | Get Music Byte |
| 21 | FA 80 | ANI 80 | 10 | AND 80 with B9;oct.byte = 10 |
| 23 | 32 2A | BZ | 10 | If oct.byte is 00, go to 2A |
| 25 | 8E | GLO E | 33 | Get low oct. freq. constant |
| 26 | F6 | SHR | 19 | Shift rt.,nominally halves 'E' |
| 27 | FF 01 | SMI 01 | 18 | Final hi-oct.freq.constant mod'n |
| 29 | AE | PLO E | 18 | Store hi-oct.freq.constant |
| 2A | C4 | NOP | | Spare.(See "New Uses") |
| 2B | 9B | GHI B | 30 | Get duration constant |
| 2C | BF | PHI F | 30 | Make a copy.(See "New Uses") |
| 2D | 9F | GHI F | 30 | Get duration constant |
| 2E | 32 04 | BZ | 30 | If hi-F=00,get next Music Byte |
| 30 | 8E | GLO E | 18 | Get frequency constant |
| 31 | FF 01 | SMI 01 | 17 | Decrement freq.const;hold in 'D' |
| 33 | 2F | DEC F | | Decrement duration constant |
| 34 | 3A 31 | BNZ | 17 | Back thru' loop until 'D' = 0 |
| 36 | 8C | GLO C | 09 | Get note constant |
| 37 | 32 2D | BZ | 09 | If 00, back thru' dur'n loop |
| 39 | 31 3E | BQ | | If Q = 1, go to reset Q |
| 3B | 7B | SEQ | | Set Q |
| 3C | 30 2D | BR | | Loop back until dur'n ends |
| 3E | 7A | REQ | | Reset Q |
| 3F | 30 2D | BR | | Loop back until dur'n ends |
| 40 | | | | 2D also used as 'rest' counter |
| 41 | C4 | NOP | | Spare; end of main program. |
| 42 | FF | | | (See "New Uses") |
| 43 | C4 | NOP | | Reserved loc'n for speed mod'r |
| 44 | 57 | A# (or B♭) | | 'Note' frequency constants |
| 45 | 49 | C# (or D♭) | | |
| 46 | 41 | D# (or E♭) | | |
| 47 | 36 | F# (or G♭) | | |
| 48 | 30 | G# (or A♭) | | |
| 49 | 33 | G | | |
| 4A | 5C | A | | |
| 4B | 52 | B | | |
| 4C | 4D | C | | |
| 4D | 45 | D | | |
| 4E | 3D | E | | |
| 4F | 39 | F | | |
| 50 | ( ) | | | 1st Music Byte of melody |

# RS-232C INTERFACE

Bernie Murphy

If you are fortunate enough to have an RS-232 device such as a terminal, digital cassette deck, et., and wonder how to plug that funny 25 pin connector into your micro, this article should be of interest to you.

By using the circuit in figure 1, I was able to connect a 300 BPS ASCII terminal to my TEC-1802. The CD4049 was included mainly as a buffer, as I was a little nervous having a chip that has +/- 12 volts on its input line tied directly to the 1802 CPU! Perhaps a 10K pullup resistor should exist between the MC1489L and the CD4049 - my interface works fine without it.

The MC1489L is a RS-232-C line receiver and the MC1488L is the matching line driver chip. These chips are easy to use and only cost $1.75 each. There are actually 4 receivers/drivers in each chip so that other devices can use the free sections. If you are not using all inverters in the CD4049, you should tie all input lines to ground.

In order to test out the interface, the program found in listing 1 was written. Note that the input and output routines are written as subroutines, so that they can be "lifted" for other programs. If your system clock is running at a speed other than 1 MHZ, then the constants in statements 73 and 104 will have to be adjusted accordingly. The values in the program listing are for 300 BPS, and can be changed to run at other speeds if required.

References:


1. User manual for the CD1802 Cosmac Microprocessor, MPM201A, RCA Corporation, p. 73.
2. Application note for MC1489, DS9173, Motorola Semiconductor Products Inc.
3. Application note for MC1488L, DS9162, Motorola Semiconductor Products Inc.
4. EIA Standard RS-232-C, Electronic Industries Association, Washington, D.C. 20006.
5. Lancaster 'SERIAL INTERFACE', Byte Magazine, September 1975, p. 22.
6. RCA COS/MOS, SSD-203C, RCA Corporation.

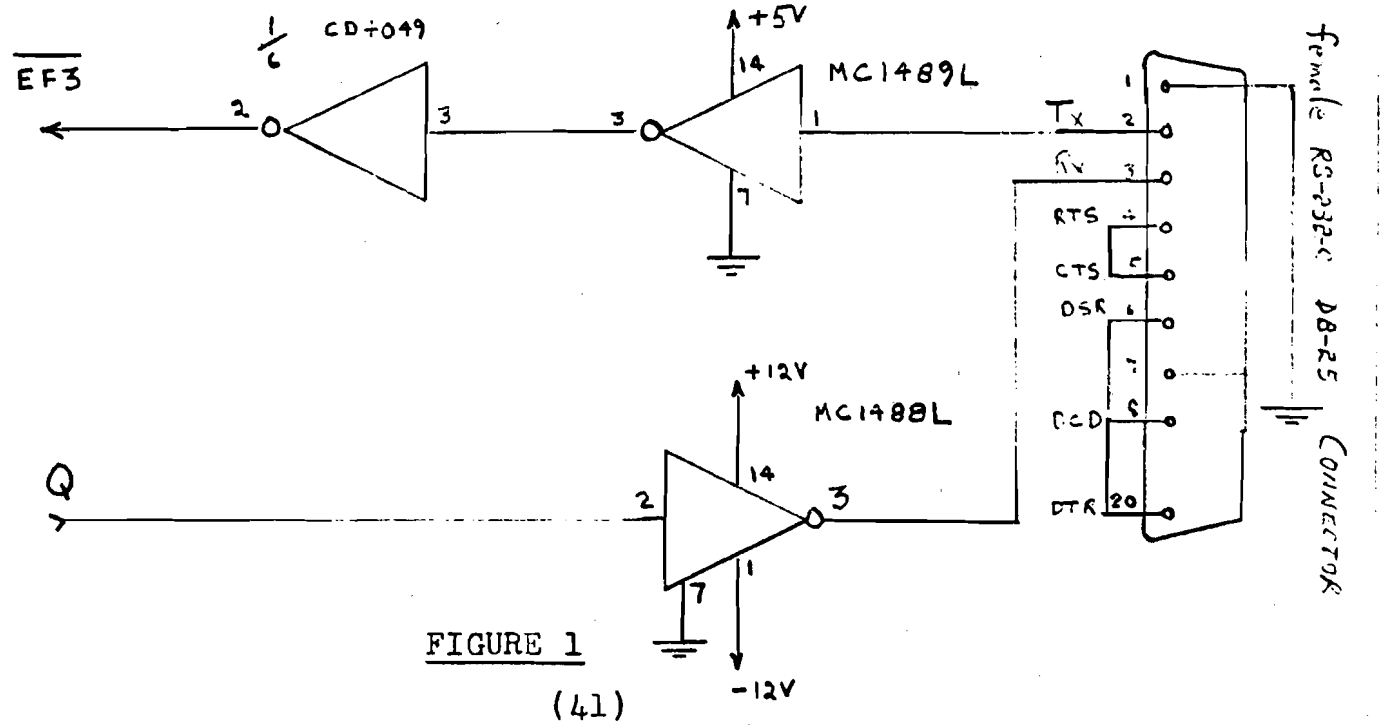




FIGURE 1

```
LOCN OBJ CODE  STMT    SOURCE STATEMENT                          1802 VER 1.3

                 1 * * * * * * * * * * * * * * * * * * * * * * * * * * *
                 2 *
                 3 * ASCIITST: TEST PROGRAM
                 4 *       THIS MAINLINE PROGRAM ECHOS CHARACTERS
                 5 *       ENTERED BY THE KEYBOARD
                 6 *
                 7 *         REGISTER USAGE:       R0=INITIAL P.C.
                 8 *                               R2=STACK
                 9 *                               R3=MAINLINE P.C.
                10 *                               R9=TX/RX SUBR P.C.
                11 *                               R15=DELAY P.C.
                12 *
                13 * * * * * * * * * * * * * * * * * * * * * * * * * * *
0000 E2         14           SEX   R2              STACK IS R2
0001 F8 00      15           LDI   0               D=0
0003 B2         16           PHI   R2              CLEAR R2 HIGH
0004 B3         17           PHI   R3              CLEAR R3 HIGH
0005 B9         18           PHI   R9              CLEAR R9 HIGH
0006 BF         19           PHI   R15             CLEAR R15 HIGH
0007 F8 11      20           LDI   MAINLOOP        ADDR OF TESTLOOP
0009 A3         21           PLO   R3              P.C. FOR MAINLINE
000A F8 FF      22           LDI   $FF             STACK ADDR FOR NOW
000C A2         23           PLO   R2              SAVE IN R2,
000D F8 39      24           LDI   DELAY           SET UP R15 AS DELAY
000F AF         25           PLO   R15             SUBROUTINE P.C.
0010 D3         26           SEP   R3              SET P.C. TO R3
0011 F8 43      27 MAINLOOP  LDI   RXENTRY         SET UP R9 AS RX
0013 A9         28           PLO   R9              SUBROUTINE P.C.
0014 D9         29           SEP   R9              CALL RX ROUTINE
0015 52         30           STR   R2              STORE CHAR ON STACK
0016 64         31           OUT4                  DISPLAY CHAR
0017 22         32           DEC   R2              BECAUSE OF OUT4
0018 F8 20      33           LDI   TXENTRY         SET UP R9 AS TX
001A A9         34           PLO   R9              SUBROUTINE P.C.
001B 02         35           LDN   R2              GET CHAR FROM STACK
001C D9         36           SEP   R9              CALL TX ROUTINE
001D 30 11      37           BR    MAINLOOP        DO IT AGAIN
                38 * * * * * * * * * * * * * * * * * * * * * * * * * * *
                39 * TXTTY: TTY OUTPUT ROUTINE
                40 *
                41 *         REGISTER USAGE :      R15=PC FOR DELAY SJBR
                42 *                               R14=BIT COUNTER
                43 *                               R13=CHAR INPUT
                44 *                               R12=LOOP COUNTER
                45 *                               R9=PC FOR TXTTY
                46 *
                47 *         INPUTS:               D=CHARACTER TO BE
                48 *                               SENT
                49 *
                50 *         OUTPUTS:              Q LINE...SERIAL OUT
                51 *
                52 * * * * * * * * * * * * * * * * * * * * * * * * * * *
001F D3         53 TXRETURN SEP   R3              RETURN TO CALLER
0020 AD         54 TXENTRY  PLO   R13             SAVE CHAR IN D
0021 7B         55          SEQ                   SET MARK (ON)
0022 F8 08      56          LDI   8               8 BITS PER CHARACTER
0024 AE         57          PLO   R14             NUMBER OF BITS TO GO
0025 7A         58          REQ                   SET START BIT
```

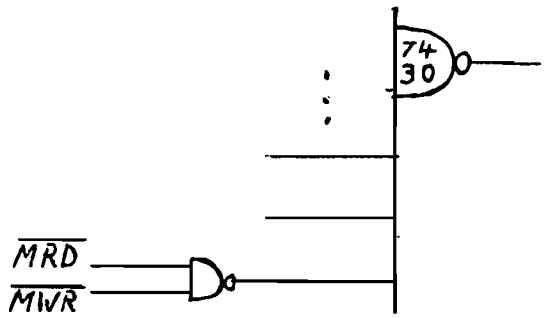
```
LOCN OBJ CODE  STMT    SOURCE STATEMENT                          1802 VER 1.3

0026 DF         59          SEP   R15             DELAY ONE BIT TIME
0027 8D         60 TXSHIFT  GLO   R13             GET BITS TILL NOW
0028 76         61          SHRC                  GET HIGH BIT INTO DF
0029 AD         62          PLO   R13             SAVE SHIFTED VALUE
002A CF         63          LSDF                  TEST IF BIT WAS THERE
002B 7A         64          REQ                   NO...SO SET SPACE
002C 38         65          SKP                   SKIP THE SEQ
002D 7B         66          SEQ                   SET TO MARK
002E DF         67          SEP   R15             DELAY ONE BIT TIME
002F 2E         68          DEC   R14             NUMBER OF BITS TO GO
0030 8E         69          GLO   R14             INTO D
0031 3A 27      70          BNZ   TXSHIFT         BRA IF NOT DONE YET
0033 7B         71          SEQ                   SET STOP BIT (MARK)
0034 DF         72          SEP   R15             DELAY FOR 1 STOP BIT
0035 DF         73          SEP   R15             ANOTHER ONE...
0036 30 1F      74          BR    TXRETURN        RETURN TO CALLER
                75 *
0038 D9         76 DLYRTN   SEP   R9              BACK UP 1 LEVEL
0039 F8 3C      77 DELAY    LDI   60              1 BIT TIME AT 1MGHZ
                78 * NOTE:  ABOVE VALUE WORKS ONLY FOR 1 MGHZ CLOCK
003B AC         79          PLO   R12             SAVE AWAY
003C 2C         80 DECR     DEC   R12             COUNT DOWN
003D 8C         81          GLO   R12             RESULT INTO D
003E 3A 3C      82          BNZ   DECR            DONE ?
0040 30 38      83          BR    DLYRTN          RETURN TO CALLER
                84 *
                85 * * * * * * * * * * * * * * * * * * * * * * * * * * *
                86 * RXTTY: TTY INPUT ROUTINE
                87 *        USES FLAG 3 AS SERIAL INPUT PORT
                88 *
                89 *        REGISTER USAGE :      R15=PC FOR DELAY SJBR
                90 *                              R14=CHAR,BIT COUNTER
                91 *                              R9=PC FOR RXTTY
                92 *
                93 *        INPUTS:               FLAG EF3
                94 *
                95 *        OUTPUTS:              RESULT IN D REG
                96 *
                97 * * * * * * * * * * * * * * * * * * * * * * * * * * *
0042 D3         98 RXRETURN SEP   R3              RETURN TO CALLER
0043 F8 00      99 RXENTRY  LDI   0               ZERO D REG
0045 BE        100          PHI   R14             CLEAR R14 HIGH
0046 F8 08     101          LDI   8               EXPECT 8 BITS
0048 AE        102          PLO   R14             NUMBER OF BITS TO GO
0049 36 49     103 RXSTART  B3    *               WAIT FOR START BIT
0048 F8 1E     104          LDI   30              1/2 BIT TIME
               105 * NOTE:  ABOVE VALUE WORKS ONLY FOR 1 MGHZ CLOCK
004D 1F        106          INC   R15             15=R15+2 TO MISS
004E 1F        107          INC   R15             LDI 60 AT 'DELAY'
004F DF        108          SEP   R15             DELAY 1/2 BIT TIME
0050 36 49     109          B3    RXSTART         GOOD START BIT?
0052 DF        110 RXLOOP   SEP   R15             DELAY ONE BIT TIME
0053 FC 00     111          ADI   0               RESET DF
0055 3E 59     112          BN3   RXBITON        BIT ON
0057 FF 00     113          SMI   0               SET DF
0059 9E        114 RXBITON  GHI   R14             GET CHAR
005A 76        115          SHRC                  SHIFT DF INTO CHAR
0058 BE        116          PHI   R14             SAVE RESULT TILL NOW
```

```
LOCN OBJ CODE  STMT    SOURCE STATEMENT                    1802 VER 1.3

005C 2E          117          DEC   R14          NUMBER OF BITS TO GO
005D 8E          118          GLO   R14          INTO D
005E 3A 52       119          BNZ   RXLOOP       BRA IF NO DONE YET
0060 DF          120          SEP   R15          DELAY FOR STOP BIT
0061 9E          121          GHI   R14          RESULT INTO D
0062 30 42       122          BR    RXRETURN     RETURN TO CALLER
                 123 *
0064             124          END

   0 DIAGNOSTICS GENERATED
  12 SYMBOLS

SYMBOL TABLE:
MAINLOOP    0011          DECR        003C
TXRETURN    001F          RXRETURN    0042
TXENTRY     0020          RXENTRY     0043
TXSHIFT     0027          RXSTART     0049
DLYRTN      0038          RXLOOP      0052
DELAY       0039          RXBITON     0059
```

## ERRATA: MEMORY-MAPPED I/O

Tom Crawford

One change is required in the schematics shown in Fig. 2 and 3 on pages 17 and 18 of Issues #4 of IPSO FACTO. The change is the same for both schematics, and is shown below. It consists of replacing TPB at one input of the 7430 IC with the NAND combination of $\overline{MRD}$ and $\overline{MWR}$. The reason for this is not obvious until one examines the internal structure and operation of the 1802 during the Non-memory cycle of Group 1 type instructions. These instructions generally operate on the 1802's internal registers, and in so doing the content of the indicated register is transfered to the internal Address register, and thence to the Address buss, during the Non-memory cycle (state S1). If TPB is used as a memory-mapped I/O strobe, then under certain conditions, there will appear to be a memory access during a Non-memory cycle! This can be prevented through the use of $\overline{MRD}$ and $\overline{MWR}$ to generate the strobe, since neither of these will go low during a Non-memory cycle.




Another advantage of this change is that the I/O strobe is now wider than it was with TPB. This will allow faster CPU operation with certain critical items of hardware, such as UART's. (The one I am using has a minimum data strobe time of 500 nS, restricting CPU clock frequency to 2 MHz with TPB-based strobe, and to 4 MHz with $\overline{MRD}$ - $\overline{MWR}$ - based strobe.)

## ERRATTA: 1802 INTERRUPT PROCESSING

Jim Howell

Dear Tom:

I just received the January issue (#4) of IPSO FACTO, (having recently sent in my membership), and would like to comment on the article about 1802 interrupts (page 28). Contrary to what is stated in the article, there is no problem if X is set to 3 (or any other value) when an interrupt occurs. An interrupt causes the current values of P and X to be saved within the processor (in register T), and then P is set to 1 and X is set to 2. No memory reference occurs during the interrupt cycle. The beginning of the interrupt routine must save register T using the SAV instruction. (You can start the interrupt routine with a SEX R2 instruction if it will make you feel better, but it's not necessary.)

My computer is a COSMAC ELF II. I have on order an extender board from Quest. This board, designed to go with their "Super ELF", has 4K of memory and a cassette interface and a monitor in PROM. I have a manual paper tape reader and a TV typewriter and an 8K memory board, none of which are attached to my ELF yet, but they will be soon, hopefully.

I have written a few game programs which are short enough to be entered into memory from the hex keyboard. I have also written (and hand-assembled) a "semi-tiny BASIC" interpreter. "Semi-tiny BASIC" occupies 4K of memory, has 26 variables (A to Z), and uses 2-byte integer arithmetic. At present it has not been debugged since my computer does not yet have enough memory or peripherals.

Jim Howell

Jim:

As you stated in your letter, the COSMAC 1802 does indeed set X = 2 when interupt servicing is initiated. I assumed (incorrectly) that X remained unchanged but your letter and re-reading the COSMAC Product Guide, MPG-180, page 6 has got my thinking correct. Thank you! Bernie Murphy

(Ed. note: When you get your BASIC going please let us know. How about a write up on that Quest board?)

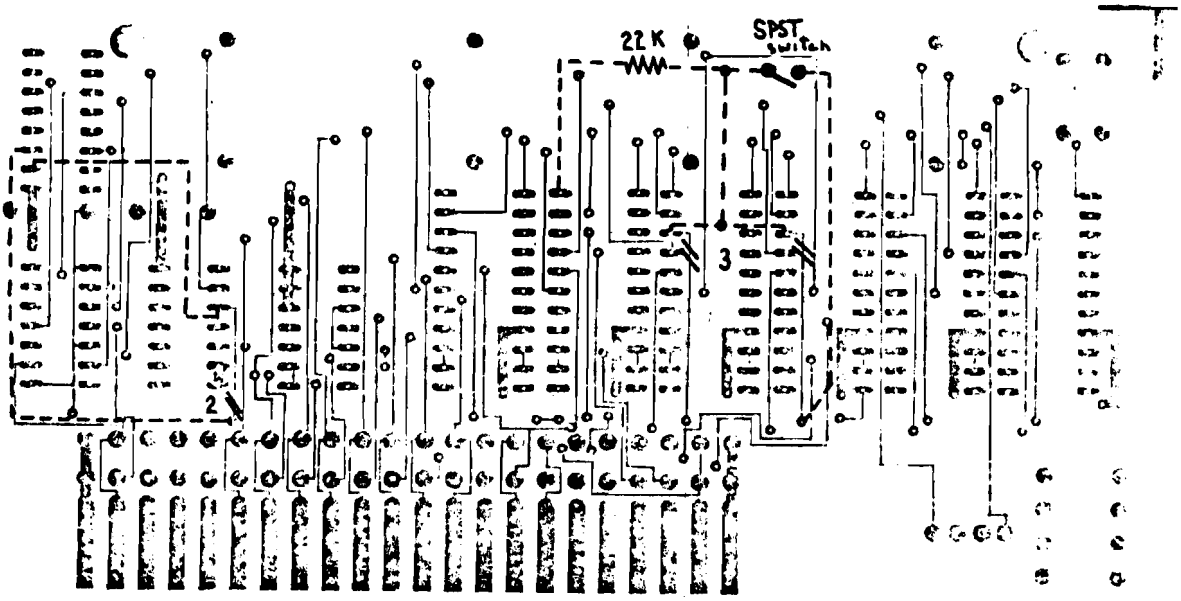
## TEC 1802 MB1 (0.75 K MEMORY BOARD)

Dan Carrigan
TEKTRON Equipment Corp.

A. SPEED IMPROVEMENT

To improve the speed of operation of this memory board there are 3 basic modifications (see figure 1):

1. Invert TPA. TPA is used to clock the 4013 flip flop to latch in the high order address lines. One of the unused inverters in the 4049 IC can be used to invert TPA. The inverted signal, TPA is used to clock the 4013 IC. Since the flip flop is edge triggered on the positive rising edge, this modification allows more set up time for the high order address lines, allowing them to be properly latched at higher speeds.
2. Change the 4013 IC (IC2) to a Fairchild F4013. This part is capable of performing at higher speed than either the Motorola or RCA part.




----- Jumper

## FIGURE 1:

**A. SPEED IMPROVEMENT**

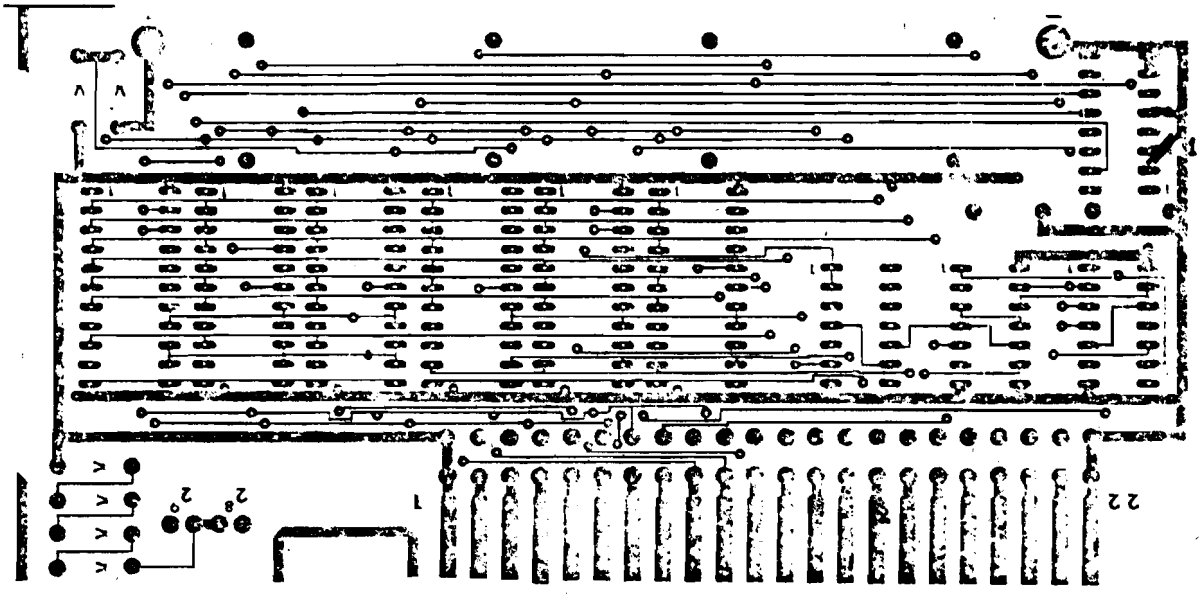
Cut foil at points 1 & 2

Place jumpers as shown

**B. MEMORY PROTECT**

Cut foil at point 3

Wire switch as shown



3. Although not necessary, the following modification can add another 5-10% speed improvement. By placing a 3.9K series resistor in the signal path of TPA to the 4049 IC (inverter) a further delay is introduced. This allows more set up time for data (high order address lines) to be latched by the 4013 latch. The combination of resistance and input capacitance to the 4049 IC gives a further delay of approximately 60 nsec.

B. TEC 1802 SPEED CONSIDERATIONS

This first set of data applies to the TEC 1802 operating with the 0.75 K memory board (MB1). Without modifications the board will operate properly up to video compatible speed (RCA 1861 at 1.76MHz) but with the modification discussed previously consider the following.

| PART (Function) | ACCESS TIME (Propagation delay) | |
|---|---|---|
| | TYPICALLY | MAX |
| 5101 CMOS RAM CE2 to output | ——— | 1100 nSec |
| MC14071B | 160 nSec | 320 nSec |
| F4013PC | 95 nSec | 170 nSec |
| MC14049B | 80 nSec | 160 nSec |

RCA specifies the memory access time to be 5.5 clock cycles after the trailing edge of TPA. Using the above table of data to calculate the maximum clock frequency gives:

Total delay = 1100 + 160 + 95 + 80 = 1435 nSec

$$F_{max} = \frac{5.5}{1435\ nSec} = 3.83\ MHz$$

When the small memory board (MB1) is used with the large expansion board (MB2) a further delay must be considered since the large board enables the small one. The delay after TPA that enable the MC14071 on the small board is typically 300 nsec.

Therefore the maximum clock frequency becomes

$$F_{max} = \frac{5.5}{(1100+300+160)\ nsec} = 3.52\ MHz$$

Aside from the actual chip access time of the RAM memory the main problem in improving the speed is with latching in the high order address lines. (See Figure 2) With 5 Volt operation at 3.3 MHz these high order address lines are available for typically 300 nsec. By using a split power supply this length of time can be increased. With the RCA 1802 operating with a split supply at 7 volts the high order memory address lines double their width. This allows the system to be run at higher speeds. Since RCA guarantees the 1802 for 3.2 MHz at 5 volt and 5 MHz on a split supply of 5 and 10 Volts

any speed attained above those indicated are dependant on the individual CPU chip. However the D chip has been seen to operate on a single 5 volt supply at 4.0 MHz and even higher.

If speed is needed then the split power supply is the way to go. The split supply increases the length of time the high order address lines are available, making them easier to latch.

The optimum voltage for highest speed is around 7.5 Volts. This is because the width of TPA actually gets narrower as the voltage exceeds 8 volts.

SPLIT SUPPLY OPERATION

Pin 40 on the 1802 is disconnected from the Vcc bus on the TEC-1802 PCB and is connected to a low power +7.5 Volt supply.

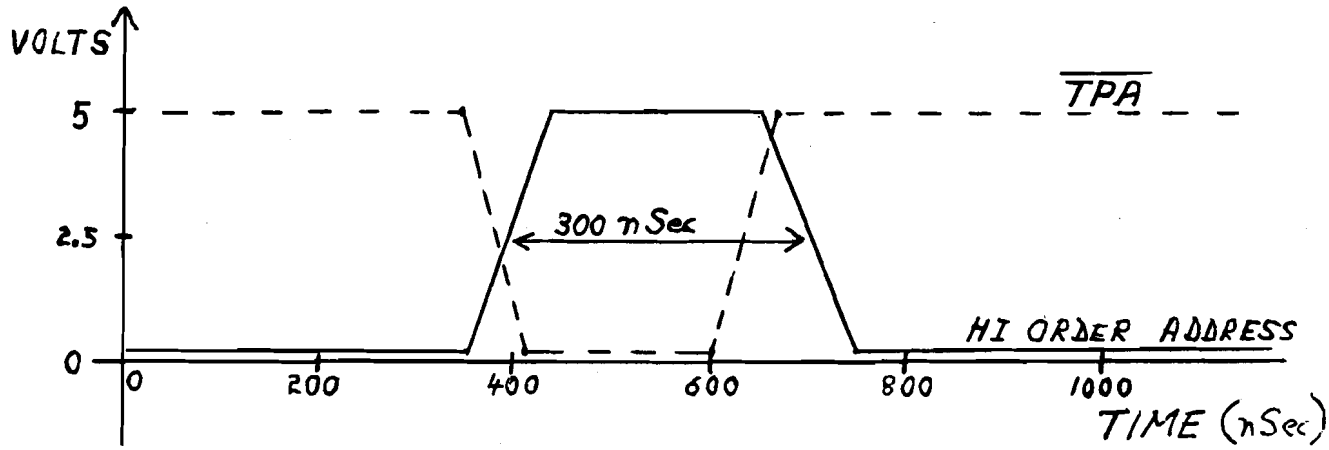



FIGURE 2 MB1: 0.75 KRAM 5 Volt Operation freq. 3.3 MHz

C. TEC 1802 MB2 (7K STATIC RAM - 2K EROM)

This board has been designed for memory expansion and compatibility with the TEC 1802 MB1 (0.75 K memory board). Using LS TTL and L TTL the board, by itself, is capable of 5 MHz operation if the CPU is used with a split power supply.

Two types of EROM can be used on this board (selected via jumpers). For single 5 volt operation the more expensive INTEL 2758 1KX8 EROM can be used. If the supply voltage of -5, +12 are available the cheaper 2708 EROM can be used.

The total power consumption using National MM2102AN-6L low power RAM (450 nsec access) and Intel B2758 EROM (450 nsec access) is approximately 1.25 Amp.


George York

In 1976 and 1977 the Hamilton Section of IEEE ran a number of Microprocessor courses based on an RCA 1802 system developed by Eugene Tekatch of Tektron Equipment Corporation. A group of the students, interested in maintaining contact and upgrading expertise in the field, formed a club in September, 1977.

Unquestionably the driving force behind the formation of the club was the efforts of 4 people--Tom Crawford, Wayne Bowdish, Claus Doerwald and Eugene Tekatch. As is the case with most organizations, these 4 people also assumed an immense extra work load. Their dedication has resulted in amazing success for the club and the newsletter. However, in the interests of preserving careers, health and marriages, not necessarily in that order, these people have to be replaced in the upcoming club elections.

The problem is to maintain a high quality, dynamic club and still not overwork individuals. The solution is the careful choice of a few key executive and a lot of volunteer working committee people.

The particular area that requires a number of "volunteer" workers is the newsletter. Maintaining a current mailing list with over 350 names and addresses can be a formidable task. We need a "volunteer" with access to a computer for storing and printing mailing labels. The problems of getting the newsletter printed, collated, stuffed into envelopes and mailed can keep 4 people occupied. Editing the newsletter is a task in itself. Another "volunteer" or two to take care of typing and diagrams would be appreciated.

The direction the club will take under a new president has yet to be determined. However, some form of software, hardware, standard membership and program co-ordination has to be maintained. In most of these cases the tasks are now handled by a committee of one. I suggest a committee of at least 2 would lighten the load and tend to keep the ideas and objectives progressing.

The purpose of this article is, therefore, to point out some of the facts of club life and to issue a call for nominations or volunteers.

As mentioned in a previous letters column, the idea of forming sub-chapters should be considered. There are probably enough members in the London area to get something going. Anyone interested in "volunteering" for their area should get in touch with the executive.

The Constitution outlines basic election regulations but leaves the mechanics of selection undefined. Because the club is still experiencing growing pains, details of the nominating and election procedure have to be developed.

The following procedures are proposed:

1. A nominating committee shall, as soon as possible, consist of 3 past presidents and be chaired by the most immediate past president. For the present, a nominating committee selected and chaired by the president, shall propose a slate of new executive.
2. The proposed slate shall be published at least six weeks prior to the election, with a proviso that further nominations must reach the nominating committee chairman in writing (note: nominating committee chairman--Tom Crawford, 50 Brentwood Dr., Stoney Creek, Ont. L8G 2W8) at least one month prior to the election.

3. The proposed slate shall consist of at least one nominee for the positions of president, secretary, treasurer, and newsletter editor plus a number of committee executive members.
4. Election and/or ratification of the proposed slate shall be carried out at the annual meeting. If there are no contested positions on the slate, ratification of the slate shall be by a show of hands of those members present. If there are contested positions, election shall be by secret ballot as per the Constitution.

In accordance with the Constitution the meeting scheduled for May 23, 1978 is defined as the Annual General Meeting. Election or ratification of the new executive will take place at that time.

Due to the remoteness of a number of our members, it would seem reasonable to limit the nominations to persons within a forty mile radius of Hamilton.

All these proposals are optimistically hoping the organizational difficulties can be overcome to meet schedules.

One further note on policy in regard to the newsletter. Members before May 31, 1978 will receive all back issues. Members after June 1, 1978 will receive only current issues from issue 7 on. After June 1, 1978 back issues (issues 1 to 6) will only be available on a cost plus basis.

## A LOW COST 8 DIGIT DISPLAY

Blair Gerrish

This article is for those of you whose microprocessor application, such as a multimeter or test instrument, does not require a sophisticated output device such as a CRT or printer. For about $17.00, including IC's, edge receptacle and a cheap four function calculator to provide a L.E.D. display you can construct a simple calculator type display for your micro. Quite often you can find a calculator on sale which will cost less than a new fully tested 8 digit display. Also watch out for the surplus display buys as they sometimes have disabled segments or non-uniform intensity levels between segments.

In order to keep the hardware and thereby the cost to a minimum the micro's software must perform the necessary multiplexing of data to the display. For anyone not familiar with display multiplexing, a short explanation. The type of display used in this project contains 8 digits with 7 digit segments and a decimal point in each digit. The anodes of common segments in each digit are connected together and the cathodes of all the segments and decimal point in a given digit are connected together. To display a digit a positive voltage is applied at the desired segment connection and then the cathode of the desired digit position is pulled to a lower voltage level than the segment and the display lights. Segment current must of course be limited to safe values. To display different data in each digit it is only necessary to select the desired segments, enable the desired digit for a short time then disable it, select the segments for the next digit, enable it and so on. The display appears to be on continuously when the cycle of data output is kept up quickly and continuously, this will be the micro's job.

CIRCUIT OPERATION

The display hardware must be told which segments and digit to enable. An OUT1 instruction is used for segment selection, FIG. 1 gives the corresponding segment selection for a '1' state on a data line. The OUT2 instruction is used to select a digit by placing a '1' level on the data line for a particular digit as shown in FIG. 2.

| DATA LINE | SEGMENT SELECTED* |
|---|---|
| 0 | A |
| 1 | B |
| 2 | C |
| 3 | D |
| 4 | E |
| 5 | F |
| 6 | G |
| 7 | D.P. |

FIG. 1

| DATA LINE | DIGIT SELECTED** |
|---|---|
| 0 | 2 |
| 1 | 3 |
| 2 | 4 |
| 3 | 5 |
| 4 | 6 |
| 5 | 7 |
| 6 | 8 |
| 7 | 9 |

FIG. 2

*see FIG. 4 for segment positions in display

**the display used has 9 positions, but position 1 is empty. H.P. type 5082 7440

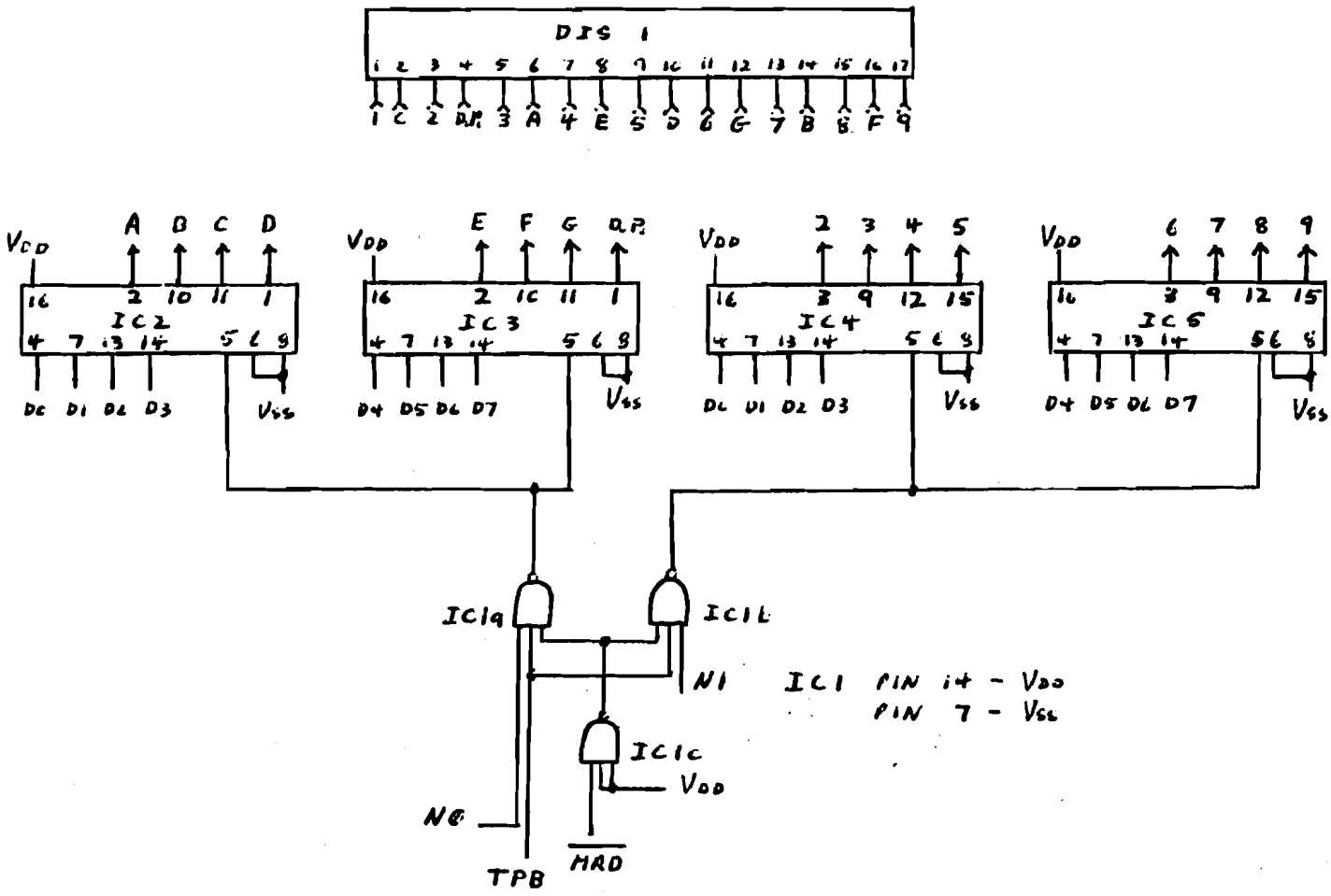
When an OUT1 is executed the data lines are valid during the time period when $\overline{MRD} \cdot TPB \cdot NO$ is true. IC1c and IC1a perform this logic and provide an inversion which causes IC2 and IC3 to latch the data lines. The non-inverting outputs of these latches will source current to the display segments. When the OUT2 instruction is executed N1 goes true along with TPB and $\overline{MRD}$. IC16 will cause IC4 and IC5 to latch the data lines. The inverting outputs of these latches will sink current from the selected digit. In a multiplexed display only 1 of these outputs will be low at a given time. For those with the TEC-1802 micro kit the pad numbers on the edge connector corresponding to the required signals are given in FIG. 3.

| PAD | SIGNAL | PAD | SIGNAL |
|---|---|---|---|
| 1 | VDD | 14 | D2 |
| 8 | $\overline{MRD}$ | 15 | D1 |
| 9 | D7 | 16 | D0 |
| 10 | D6 | 22 | VSS |
| 11 | D5 | 31 | TPB |
| 12 | D4 | 41 | N1 |
| 13 | D3 | 42 | NO |

FIG. 3
TEC-1802 EDGE CONNECTOR SIGNAL LOCATION
see pg. 12 TEC-1802 MICROPROCESSOR KIT NOTES

FIG. 4

SEVEN SEGMENT DISPLAY SCHEMATIC



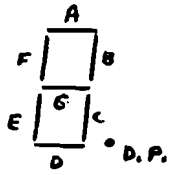

## PARTS LIST

IC1 MC14023B TRIPLE 3 INPUT NAND GATE
IC2-5 MC14042B QUAD LATCH
DIS 1 8 DIGIT 7 SEGMENT COMMON CATHODE L.E.D. DISPLAY HP 5082 7440 OR EQUIVALENT

## DISPLAY CONFIGURATION

DIGIT POSITIONS NUMBERED 2 - 9 LEFT TO RIGHT.

SEGMENT POSITIONS



Current limiting resistors are not required in series with each segment because the maximum output current from the latches, typically 2 ma., will not exceed the maximum rating of the display, typically 3-5 ma. average current for most displays of this type. If it is necessary to use larger displays then current buffers and resistors would most likely be needed.

## USING THE 8 DIGIT DISPLAY

Wayne Bowdish

The first part of this 2 part article described the hardware portion of an 8 digit display. In this section I will describe a simple test routine for the hardware. This routine implements an 8 digit HEX display. Future articles will contain various other applications of the display.

The test program consists of a subroutine which displays eight digits each time that it is called and a simple driver program to call the subroutine repeatedly. The 8 digits are stored in an 8 byte table right justified (ie. in the low order 4 bits of the byte). The character table may be filled with any 8 HEX digits which will then be displayed when the program is run. A listing of the program is on the last page of this article.

Since the driver routine is relatively simple, I will concentrate on the operation of the subroutine. This subroutine could be used with any special applications program that the user might write.

Since the display is software multiplexed the output subroutine must be called repeatedly to display the digits. If CHROUT is called often enough the display will appear to be on continuously. To output a digit, the following 3 outputs are required:

1. zero the digit select latches
2. set the segment select latches to the required pattern
3. set the required digit select latch

The output subroutine works as follows:

Memory locations #1B through #24 initialize some registers for the subroutine. Locations #25 through #41 form a loop which is executed 8 times, once for each digit. Locations 47 through 56 are a table of segment select bit patterns for each possible digit.

Within the loop the following 5 major functions are performed:

1. memory locations 25 through 2A place the address of the end of the 3 byte output table in the X-register. The last 2 bytes of this table will be filled in with the segment select data and the digit select bit. The first byte is always zero and is used to turn off all digits while the new segment select data is being output.
2. memory locations #28 through #2E place the digit select byte (initially #01) in the output table and then **shifts** the bit left one bit position so that it will be ready for the next digit.
3. memory locations #2F through #3B take a hex digit (from #0 through #F) and adds this value to the address of the start of the segment select table. This new address is then used to retrieve the proper segment select byte from the table.

```
                        .TITLE  HEXDSP

                ; TEST ROUTINE FOR 7 SEGMENT HEX DISPLAY

0000 30 0A              BR      START
0002 =  00 08   DIGTAB: .BLOCK  8       ; --- INSERT DIGITS TO BE
                                        ; --- DISPLAYED HERE
000A F8 18 A9   START:  +DLDI   CHROUT,R9        ; R9 = CHROUT P-COUNTER
000D F8 00 B9
0010 F8 17 AA           +DLDI   LOOP,R10        ; R10= MAINLINE P-COUNTER
0013 F8 00 BA
0016 DA                 SEP     R10
0017                    LOOP:
0017 D9                 SEP     R9              ; OUTPUT THE CHARACTERS
0018 30 17              BR      LOOP            ; GO BACK AND DO ANOTHER


                ; CHROUT - CHARACTER OUTPUT ROUTINE

001A DA         CHRRTN: SEP     R10             ; RETURN TO MAIN PROGRAM
001B F8 02 AF   CHROUT: +DLDI   DIGTAB,R15      ; R15 POINTS TO DATA TABLE
001E F8 00 BF
0021 F8 01              LDI     1
0023 BC                 PHI     R12             ; R12.HI IS DIGIT SELECT
0024 ED                 SEX     R13
0025 F8 46 AD   CHRLUP: +DLDI   DIGSEL,R13      ; R13 = END OF OUTPUT TABLE
0028 F8 00 BD
002B 9C                 GHI     R12             ; GET DIGIT SELECT BIT AND SAVE
002C 73                 STXD    R13             ;   THEN DEC. X TO POINT TO SEGSEL
002D FE                 SHL                     ; SHIFT DIGIT SELECT BIT FOR
002E BC                 PHI     R12             ;   NEXT DISPLAY DIGIT AND SAVE
002F 4F                 LDA     R15             ; MASK OFF ANY UNUSED DATA BITS
0030 FA 0F              ANI     0FH             ;   JUST IN CASE THEY ARE SET
0032 FC 47              ADI     CHRTAB          ; ADD LOW ADDR. OF SEGMENT
0034 AE                 PLO     R14             ;  SELECT TABLE AND SAVE
0035 F8 00              LDI     CHRTAB-.-8      ; GET HIGH ADDR. OF SEG. SEL. TAB.
0037 7C 00              ADCI    0               ;  AND ADD IN CARRY FROM PREVIOUS
0039 BE                 PHI     R14             ;   THEN SAVE HIGH ADDRESS
003A 0E                 LDN     R14             ; GET SEGMENT SELECT DATA AND
003B 73                 STXD                    ;  PUT IN OUTPUT TABLE
003C 62                 OUT     2               ; OUTPUT DIGIT DESELECT
003D 61                 OUT     1               ; OUTPUT SEGMENT SELECT
003E 62                 OUT     2               ; OUTPUT DIGIT SELECT
003F 9C                 GHI     R12             ;  AND CHECK FOR MORE TO DO
0040 3A 25              BNZ     CHRLUP          ; LOOP FOR NEXT IF NOT ALL DONE
0042 30 1A              BR      CHRRTN          ; RETURN IF ALL DONE

0044 00         DIGBLK: .BYTE   0               ; DIGIT BLANK CODE
0045 00         SEGSEL: .BYTE   0               ; SEGMENT SELECT CODE
0046 00         DIGSEL: .BYTE   0               ; DIGIT SELECT BIT
                        .SLIST
```



```
                ; TABLE TO SELECT CORRECT SEGMENTS TO BE DISPLA
                ;       NOTE SET BIT = SEGMENT ON

0047                    CHRTAB:
0047 3F                 .BYTE   #3F     ; 0
0048 30                 .BYTE   #30     ; 1
0049 5B                 .BYTE   #5B     ; 2
004A 4F                 .BYTE   #4F     ; 3
004B 66                 .BYTE   #66     ; 4
004C 6D                 .BYTE   #6D     ; 5
004D 7D                 .BYTE   #7D     ; 6
004E 07                 .BYTE   #07     ; 7
004F 7F                 .BYTE   #7F     ; 8
0050 67                 .BYTE   #67     ; 9
0051 77                 .BYTE   #77     ; A
0052 7C                 .BYTE   #7C     ; B (SMALL)
0053 39                 .BYTE   #39     ; C
0054 5E                 .BYTE   #5E     ; D (SMALL)
0055 79                 .BYTE   #79     ; E
0056 71                 .BYTE   #71     ; F

                =  00 0A        .END    START
```

4. locations #3C through #3E output the 3 bytes from the output table.
5. locations #3F through #43 check the digit select byte. If it is zero (ie. the select bit has been shifted out) then all 8 digits have been output and the subroutine returns to the calling program, otherwise the loop is repeated for another digit.

APPLICATIONS AND EXTENSIONS

The display hardware and software forms a general purpose output device. Some possible applications that come to mind are:

- clock display (hh-mm-ss and mm-dd-yy)
- multi station thermometer display (indoor outdoor etc.)
- games of course (winning ticket number etc.)
- monitor data display (address, data, function)
- test gear (volt, ohm, current, frequency, etc. display)
- and on and on and on .....

Some possible extensions are:

- decimal point selection (high order data bit set maybe)
- additional characters (ie. H, J, L, o (small), P, U and some strange ones ⊦ ⊣ = ≡ ıı ⅁ ⅂ )

Extra characters could easily be added by using the low order 5 bits of the data byte and expanding the character table out to 32 bytes.

The Association of Computer Experimenters

Minutes of Club Meeting No. 78-2

Held at Stelco Wilcox St. Auditorium

9 February, 1978 8:00 P.M.

78-2-1 The regular meeting was preceded by a 1 hour tutorial.

78-2-2 The Minutes of the last meeting were included in the Newsletter. There were no copies available for distribution therefore the Minutes were not approved.

78-2-3 George York, Secretary-treasurer, reported a current bank balance of $1,000.38.

78-2-4 Claus Doerwald, Membership Co-ordinator, announced a paid up membership of 233, with about 10 per week still coming in. There was some discussion on the possibility of cutting off back-issues for new members.

78-2-5 Wayne Bowdish, Software Co-ordinator, had no report.

78-2-6 Hardware. Eugene Tekatch, Hardware Co-ordinator was absent--no report.

78-2-7 Newsletter Editor, Tom Crawford, reported the Newsletter, Issue 4, has gone for printing. We now require about 400 copies to cover our needs e.g. members plus back issues for new members.

78-2-8 Education Co-ordinator, Ken Smith, announced the next tutorial session will be February 23, 8:00 P.M. at the Stelco Wilcox Auditorium.
He mentioned Adam Osborne & Associates are supposed to be coming out with a book on Cosmac programming.

78-2-9 New Business--There was no report on possible alpha-numeric I/O kits.
Anyone with info on used terminals or printers should get in touch with a member of the executive.
There will be an article in the Hamilton Spectator on home microcomputers that should mention the club.
There was some discussion on the development of small systems and/or communicating between CPU's.
Some members volunteered to do a literature survey and a short synopsis for the Newsletter of any particularly good articles. The members involved were Jeffe Waite-Kilobaud, Ed Leslie-Byte, Paul Birke-Dr. Dobbs, Blair Gerrish-Popular Electronics, John Williams-Radio Electronics.



cont'd....

78-2-9 (cont'd)
The members would like a cassette standard for the next meeting, if possible. There was some discussion on the pros and cons of hardware, software, Kansas City, etc. approaches.

The members were reminded that the election of a new club executive should be considered. As outlined in the Club Constitution, the elected positions are President, Secretary/Treasurer and Newsletter Editor. Executive-at-large members are also required to fill positions at the discretion of the President.
The Constitution, specifically Article 5.4 requires amendment to allow the executive committee to function in the absence of the president. The original article reads:

Article 5.4 Meetings of the executive committee shall be at the call of the president.

The amended article would read:

Article 5.4 Meetings of the executive committee shall be at the call of the president or, in his absence, the secretary/treasurer.

This Constitutional amendment will be voted on at the first meeting after the Newsletter Issue 5 is mailed.

The meeting schedule for meetings and tutorials until the end of May is published in Newsletter Issue 4.

78-2-10 Motion to adjourn meeting
Proposed - Jeffe Waite
Seconded - Doug Inkster
Passed

The meeting adjourned at 10:10 P.M.
About 50 people attended the meeting.

The Association of Computer Experimenters

Minutes of Club Meeting No. 78-3

Held at Stelco Wilcox St. Auditorium

7 March, 1978 8:00 P.M.

78-3-1 The regular meeting was preceded by a 1 hour tutorial.

78-3-2 Motion to adopt Minutes 78-1 and 78-2
Proposed - Walt Smith
Seconded - Ed Benvenuiti
Carried unanimously

78-3-3 George York, secretary-treasurer, reported the bank account was being changed to the new club name. The last statement showed a current balance of $900.38.

There was some discussion on the need to cover the club's liabilities (eg. incorporating, co-op or insurance).

Motion - the executive is authorized to investigate the liabilities aspect of the club. The investigation can include a nominal legal fee.
Proposed - Walt Smith
Seconded - Brian Fox
Carried unanimously.

It was noted the Burlington Amature Radio Club has just finished incorporating. Their executive should be consulted.

78-3-4 Membership. There are now 323 paid members.

78-3-5 Software - No report

78-3-6 Education. Ken Smith took a quick survey to find what subjects would be of interest for the next tutorial sessions. The suggestion was for a keyboard and CRT interface hardware and software. Group effort will be encouraged in the tutorial sessions.

78-3-7 The info sheet for Lyle Sandy on the back of issue #4 should be filled out and sent in.

78-3-8 Motion - the club should pay for a 1 year subscription to Byte and Dr. Dobbs in the name of the Hamilton Public Library. These magazines would be available at the Hamilton Reference Library.
Proposed - Tom Crawford
Seconded - Jim Rix
Carried

Note that the Hamilton Public Library is getting a subscription to Kilobaud magazine also.

cont'd...

78-3-9 Ken Collins has volunteered to give a wire wrapping demonstration at the next meeting.

78-3-10 Hardware Co-ordinator, Eugene Tekatch reported there will be a software 300 baud K.C. interface as a club standard. There will also be a 1200 baud pulse duration software interface developed.

Tektron will soon have a board developed with an 1861 video chip with RF interface, cassette and teletype interface plus an n-line decoder on board.

Tektron's 7K board with 2K EROM is now available.

Large display costs are still excessive but, with the cost of chips (eg. character generators) dropping, a cheap large display may soon be possible.

Gene mentioned that there are a large group of the membership who are interested in and need more help with more hardware applications. A call was issued for ideas and/or circuits for more hardware applications.

Some ideas suggested were: an automatic telephone dialer, an indoor/outdoor temperature display, an 8 digit display.

78-3-11 Motion to adjourn meeting
Proposed - Brian Fox
Seconded - Bert DeKat
Passed

The meeting adjourned at 10:10 P.M.
About 50 people attended the meeting.

## CLUB NOTES

### SYSTEM INFO SHEETS

Lyle Sandy

Fill out your system information sheet now....! Returns have been slow (17 so far) to the information sheets included with the last newsletter. Please respond, even if you have only the simplest system, or even no system at all yet. It is important that your aspirations, as well as accomplishments, be made known to other club members in order that good technical progress can be made. Fill out the form and get it in the mail ASAP so that all info can be summarized in the next newsletter.

### LETS PUT THE 1802 TO WORK

Eugene Tekatch

Well its about time we put the 1802 to work. We have seen an abundance of assemblers, monitors, editors, etc. but the majority of ACE members have not got envolved in microprocessor applications. In asking myself why this is the case, I have determined what I think is the reason. My micro course moved very fast through basic theory & programming, leaving very little time for hardware applications. Without demonstrations & working system doing hardware functions (driving a relay, controlling a motor, high voltage hook-up, TTY connection, CRT, etc.) an air of mystery still existed and in turn people were reluctant to try applications.

I have proposed that the next issue of the newsletter be a special project edition. Send in the data on your working projects BIG or SMALL including hardware and software. Mail as soon as posible to the Editor of this Newsletter, or to myself:
EUGENE TEKATCH, TEKTRON EQUIPMENT CORP., 263 BARTON ST. UNIT #19
STONEY CREEK, ONTARIO L8E 2K4

Lets share ideas and in turn encourage more projects.

### MEETING TIMES

Claus Doerwald

We've had numerous requests from local members to change the date of the regular club meetings. I'm afraid that for the present we're locked into the schedule on pg. 35 of issue #4 of the newsletters, but in the future we'd like to optimize the schedule. We need your help in the form of a return of this questionaire:

MEETING SCHEDULE SURVEY

If you plan to attend club meetings please indicate preferred nights in the order of priority (1,2,3, etc. with the highest being one; if two or more evenings have equal priority use the same #)

| | | | |
|---|---|---|---|
| MONDAY | ☐ | THURSDAY | ☐ |
| TUESDAY | ☐ | FRIDAY | ☐ |
| WEDNESDAY | ☐ | SATURDAY | ☐ |

Mail to: CLAUS DOERWALD, c/o DOFASCO ENGINEERING DEPT. P.O. BOX 460, HAMILTON, ONTARIO L8N 3J5

This page contains a change of address form and an application for membership form. If your address is incorrect or if your address has changed, please mail in the change of address form. This will ensure that you get your next copy of the IPSO FACTO Newsletter. Don't forget to return your old mailing lable. This will allow us to find the incorrect address and correct it.

If you know of anyone who would be interested in joining our club, why not give him/her the membership application. We are constantly looking for new members with new and interesting ideas. Make cheques payable to the Association of Computer Experimenters.

---

Association of Computer Experimenters

CHANGE OF ADDRESS FORM

. ATTACH MAILING LABLE HERE

FIRST NAME

LAST NAME

FIRST LINE OF ADDRESS

SECOND LINE OF ADDRESS

CITY & PROVINCE

POSTAL CODE

---

MEMBERSHIP APPLICATION FORM FOR THE Association of Computer Experimenter

FIRST NAME

LAST NAME

FIRST LINE OF ADDRESS

SECOND LINE OF ADDRESS

CITY & PROVINCE

POSTAL CODE

| LOC. | CODE | LABEL | OPCODE | OPERAND | COMMENT |
|---|---|---|---|---|---|
| | | | | | |